引力搜索算法及其应用

刘勇　马良　著

上海人民出版社



上架建议：管理学

ISBN 978-7-208-12544-5



定价: 38.00元

易文网: www.ewen.co

引力搜索算法及其应用

刘勇　马良　著

上海人民出版社

图书在版编目(CIP)数据

引力搜索算法及其应用/刘勇,马良著.—上海:
上海人民出版社,2014
ISBN 978-7-208-12544-5

Ⅰ.①引… Ⅱ.①刘… ②马… Ⅲ.①最优化算法-
研究 Ⅳ.①O242.23

中国版本图书馆CIP数据核字(2014)第212137号

责任编辑 曹怡波
封面设计 张志全



引力搜索算法及其应用
刘 勇 马 良 著
世纪出版集团
上海人民出版社出版
(200001 上海福建中路193号 www.ewen.co)
世纪出版集团发行中心发行 上海商务联西印刷有限公司印刷
开本720×1000 1/16 印张8.5 插页2 字数131,000
2014年10月第1版 2014年10月第1次印刷
ISBN 978-7-208-12544-5/O·1
定价38.00元

前　　言

他山之石，可以攻玉。

——《诗经·小雅·鹤鸣》

目前，由于现实中的优化问题越来越复杂，大规模、高维数、多目标、非线性和不可微已成为其基本特征，用传统优化算法进行求解已越来越困难。而智能优化算法在使用时不需要梯度信息，特别适用于传统优化算法难以求解的复杂困难问题，并已在诸多领域中获得广泛的应用。

遗传算法、模拟退火算法、禁忌搜索算法、微粒群优化算法和蚁群优化算法等，都是典型的智能优化算法。但受到这些算法本身搜索机制的限制，它们在实际应用中也或多或少暴露出其固有的一些缺陷。人们一方面致力于对现有算法的不断改进，另一方面则希望设计一些新的优化算法。基于物理学原理的引力搜索算法，由于概念简单、易于实现且优化性能较好，业已引起了相关研究人员的极大关注。

本书在系统研究引力搜索算法机制原理的基础上，对所提出的引力搜索算法给出了相关的数学证明，从理论上说明了算法的可行性；探讨了引力搜索算法在连续优化和离散优化领域的若干改进策略，阐述了算法的一些典型应用，并从实验角度验证了算法的有效性；研究了引力搜索算法和其他优化算法的融合策略，分析了算法的发展方向。

本书是国内首部引力搜索算法方面的专著，致力于推动引力搜索算法的发展与应用，以及管理科学和系统科学领域的相关研究，并为计算机科学和控制科学等领域提供方法和工具。

本书的部分工作受到国家自然科学基金项目(70871081)和上海市一流学科建设项目资助(S1205YLXK)的资助，在此谨致谢意。

本书撰写过程中，许多人都给予作者以很大的帮助，在此一并表示感谢，其中，

特别感谢上海交通大学田澎教授的关心和鼓励，以及家人们多年来对作者工作的理解和支持。

限于作者的学识水平有限，书中错误和疏漏之处在所难免，恳请广大读者批评指正。

作　者

2014 年 6 月

目　　录

第三章　单目标连续优化的引力搜索算法

第四章　多目标连续优化的引力搜索算法

第五章　引力搜索算法的若干应用

第一章　智能优化算法

最优化理论和算法是一个重要的研究内容，其追求的目标是如何在众多的方案中找出最优方案。最优化问题普遍存在于我们的工作和生活中。例如，安排生产计划时，选择怎样的实施方案才能提高产值与利润；在城市建设规划时，如何安排住户、学校、医院、超市和工厂等单位的布局，才能有利于各行各业的发展，也能方便生活；工程设计时，如何确定设计参数，使得设计方案既能够满足设计指标又能够降低成本；资源分配时，如何分配有限的资源，使得分配方案既能满足各个方面的要求，又能获得好的效益。目前，优化作为一种希望实现的目标已经深入到各个领域，诸如此类，不胜枚举。最优化就是为这些问题的求解，提供理论基础和解决方法，是一门实用性强和应用广泛的学科[1,2,3]。

长期以来，人们一直对优化问题进行探讨和研究。早在17世纪，Newton发明微积分的时代，就已经提出极值问题，随后又出现了Lagrange乘数法。1847年，Cauchy提出了最速下降法。1947年，Dantzig提出了单纯形法。随着研究的深入，人们提出了很多经典的优化算法。按照优化策略的不同，传统的优化算法可以大致分为以下几类：(1)枚举法，枚举法是将整个可行解空间所有点的性能进行比较并找出其中的最优点。枚举法有完全枚举法和隐式枚举法等。算法的搜索策略最简单，但计算量也最大。枚举法适用于可行解空间是有限集合的情况。(2)解析法，解析法在优化过程中使用目标函数的解析性质，如一阶导数、二阶导数等。解析法是从一个初始点出发，并根据目标函数的梯度方向来确定下一步的搜索方向。常见的解析法有Newton法、共轭梯度法和变尺度法等[4,5]。当目标函数有多个极值点时，算法难以找到全局最优点。

传统优化算法为优化问题的求解提供了巨大帮助，但在计算过程中传统优化算法也暴露出一些缺点。这些缺点主要表现在以下几点：(1)传统优化算法通常都是从一个初始点出发，每次迭代过程中也仅仅对一个点进行计算，这种算法架构很

难发挥现代计算机高速计算的性能,尤其是高性能的多处理器的计算机和现代并行计算的模式往往很难应用到传统优化算法中,这样就限制了算法求解大规模问题的能力和计算速度。(2)传统优化算法一般都要求目标函数是连续可微的,甚至有时要求是高阶可微的,但是在实际问题中这样的条件往往很难得到满足,这样就限制了传统优化算法的应用范围。(3)传统优化算法在每一次迭代时都要求迭代向改进方向移动,即每一次都要求目标函数值有所降低(以最小值优化问题为例),这样算法就不会具有“爬山”能力。若算法陷入某个局部的低谷,则无法继续搜索该区域之外的任何其他区域。因此,算法就失去了全局搜索能力。(4)每种传统优化算法只能求解一部分问题,即算法只能求解符合其适用条件的问题。要应用某种传统优化算法往往就会简化甚至改变原有的问题,使之能满足该方法的使用条件。但是如果问题不满足任何已知的传统优化算法的适用条件,那么用传统优化算法就无法有效求解[6,7]。

考虑到传统优化算法的不足,人们希望能够设计出新的有效的优化方法。特别是进入 20 世纪 90 年代以来,所研究的实际问题的结构越来越复杂,规模越来越大,不可微、非线性、不确定性、多目标已经成为这些问题的基本特征,而建立在解析基础上的传统优化算法往往对这些问题的求解显得无能为力。因此,人们逐渐意识到,必须探索新的优化方法来解决这些问题。同时随着计算机科学与技术的飞速发展,从根本上改变了人们的工作和生活,并已经渗透到各个领域,如何充分发挥计算机技术的优势推动优化方法的发展,也是人们越来越关注的问题。

大自然永远是人类创造力的丰富源泉,生命在长期的进化过程中,积累了许多特有的功能。各种生物为了延续自身和种群的生命所展现出的智能,令人叹为观止[8,9,10]。例如,一只蚂蚁或蜜蜂的行为能力非常有限,它几乎不可能独立存在于自然世界中,但由多个蚂蚁或蜜蜂形成的群体则具有非常强的生存能力,而这种生存能力不是通过多个个体之间能力的简单叠加所能获得的。自然界的很多现象不断给人们以启示,生物和自然生态系统可以通过自身的演化使得很多在人类看来高度复杂的问题能够得到完美的解决[11-15]。受大自然的启发,人们从大自然的运行机制中找到了许多求解优化问题的新方法。从人工智能的角度,人们常常称这些算法为智能优化算法;从群体进化的特征来看,这些算法又可以称为进化计算算法;近几年人们从算法模仿自然规律的特点出发,也将这些算法称为自然计算。从应用这些方法的层面来看,方法采用什么名称并不重要,重要的是理解和掌握这

些算法所蕴含的搜索机制[6]。

智能优化算法的发展得益于运筹学、生物学、物理学、计算数学、计算机科学、人工智能和控制论等许多学科，充分吸收了这些学科的思想、概念和方法。智能优化算法，大都具备自组织性、正反馈性、鲁棒性、并行性和实现简单等特征，为在没有集中控制且不提供全局信息的条件下寻找复杂问题的求解方案提供了思路，为优化问题的求解开辟了新的手段[16—18]。

应用智能优化算法求解优化问题时，其搜索过程通常具有以下一些特征：(1)个体都具有能执行简单的空间或时间上的评估和计算能力；(2)当环境(包括其他个体)发生变化时，个体能够对变化做出响应，特别是出现值得付出代价的改变机会时，个体必须能够改变其行为模式；(3)不同的个体对环境中的某一变化所表现出的响应行为应该具备多样性[19—26]。

和传统优化算法相比，智能优化算法具有其特有的优化方式。(1)非单点操作，采用群体搜索策略。智能优化算法大都采用由多个个体组成的群体对可行解空间进行搜索，在搜索过程中能够实现对各个体所提供信息的共享。信息的传播可以避免对一些不必要区域的搜索，既能提高算法的搜索效率，又能在一定程度上避免陷入局部极值。(2)智能优化算法在求解优化问题时，目标函数和约束函数可以不必是解析的，更不必是连续和可微的。智能优化算法可有效求解那些目标函数没有明确表达式，或有表达式但不可精确估值的优化问题。此外，智能优化算法对计算中数据的不确定性有较强的适应能力。(3)智能优化算法是对自然现象和自然规律的模拟，这些现象和规律所蕴含的深刻的优化思想为算法的设计提供了坚实的科学基础。在算法中，个体行为虽然简单但通过个体之间的相互作用，整个群体涌现出智能行为，而这种行为可用于求解优化问题。(4)智能优化算法不依赖于优化问题本身的数学性质和所求问题本身的结构特征以及其他辅助信息，算法可以用于求解不同类型的优化问题，具有广泛的适用性[6,7]。

对智能优化算法的研究可以追溯到上世纪 50 年代。当时，研究人员就已经意识到可以采用 Darwin 的生物进化论来求解复杂问题。1975 年，Holland 正式提出了遗传算法，这种新发展起来的完全异于传统优化算法的方法为优化问题的求解提供了崭新的途径。由于其在求解优化问题方面的巨大潜力，许多研究人员开始对智能优化算法展开研究，并获得了显著的进展。

随着人们对大自然的进一步观察和研究，又提出了很多有代表性的智能优化

算法。例如,模拟退火算法[27]、人工神经网络[28]、蚁群优化算法[29]、微粒群优化算法[30]、人工免疫系统[31,32]、蜂群算法[33,34]和生物地理学优化算法[35,36]等。除此以外,还包括这些算法与其他方法相结合而形成的混合型算法。这些算法构成了当前优化范畴内来自跨学科的别具特色的优化策略,丰富了优化手段,为那些传统优化算法难以处理的问题提供了切实可行的解决途径[37]。但是随着研究的深入,这些算法暴露出自身存在的一些不足,如容易早熟收敛、优化精度不高和优化速度慢等问题。此外,智能优化算法的数学基础相对薄弱,没有形成系统的理论体系。这些缺点都制约了智能优化算法的进一步发展。从本质上讲,之所以会有这么多的优化算法,主要是因为到目前为止还没有找到一种方法,能够应用到所有的优化问题中。为了解决我们所面临的问题,就必须研究新的优化算法。

经过多年的发展,已经出现了很多有代表性的智能优化算法,例如人工神经网络、模拟退火算法、遗传算法、蚁群优化算法和微粒群优化算法等等。虽然这些算法的优化原理各不相同,但也具有一些共同的特征,对不同智能优化算法的理解对另一种算法的研究有着重要的启发作用。因此,这里对现有的一些智能优化算法做一些简单介绍。

1.1 人工神经网络

人工神经网络是对人类大脑系统一些特征的一种描述。简单地讲,它是一个数学模型,可以通过计算机程序进行模拟,也可以采用电子线路进行实现,是人工智能的一种研究方法[38]。1943 年,Mcculloch 和 Pitts 第一次对神经系统中的神经元进行数学建模,并提出了 MP 模型。1957 年,Rosenblatt 提出了著名的感知器模型,该模型是一个能够连续调节权值矢量的 MP 模型。1959 年,Widrow 和 Hoff 设计出了自适应线性单元的网络模型,第一次把神经网络研究从纯理论研究转向工程应用研究[39]。自此,很多学者陆续对人工神经网络展开了研究,并取得许多可喜的成果。

人工神经网络是由大量功能简单但具有自适应能力的信息处理单元——神经元按照大规模并行的方式,采用一定的拓扑结构连接而成。神经元是对人脑神经细胞功能的简化、抽象和模拟。一个人工神经网络的神经元的模型与结构描述了一个网络的输入变量转换成输出变量的过程。从数学角度看这个转换过程就是一个计算过程[40,41]。

人工神经网络发展至今,已经出现了很多版本。这里,以 Hopfield 网络为例,分析其优化策略。Hopfield 网络是一种非线性的动力学模型,采用类似 Lyapunov 函数的能量函数的概念,将神经网络的拓扑结构(采用连接权矩阵描述)和所求解问题(采用目标函数表示)对应起来,并转换为神经网络动力学系统的演化问题。所以在应用 Hopfield 网络解决优化问题之前,要将该问题映射为对应的神经网络。例如对旅行商问题的求解,首先将问题的可行解映射成一个置换矩阵,并能给出对应的能量函数,再将满足置换矩阵要求的能量函数的最小值和问题的最优解对应起来[42]。

对一般性的问题,往往需要处理以下一些工作:

(1) 选择恰当的问题表示方式,使得神经网络的输出和问题的解相对应。

(2) 设计合适的能量函数,使得其最小值和问题的最优解相对应。

(3) 根据能量函数与稳定条件设置网络参数,如连接权值与偏置参数等。

(4) 建立对应的神经网络与动态方程。

(5) 采用软件或硬件进行模拟。

目前,人工神经网络的应用已经取得了令人瞩目的进展,其应用涉及科学研究和工程应用等领域中的许多问题。例如,函数优化、组合优化、模式识别、时间序列预测、非线性系统辨识与控制、特征选择、故障诊断和信号检测等。有关人工神经网络计算机硬件的开发也已经取得显著的成果。

1.2 模拟退火算法

模拟退火算法是源于对热力学中退火过程的模拟,是一种通用的随机搜索方法,是对局部搜索方法的扩展,理论上是一种全局优化算法。1953 年,Metropolis 提出了模拟退火算法的思想。1983 年,Kirkpatrick 将模拟退火算法成功用于求解组合优化问题,才真正提出了模拟退火算法。从此,该算法受到越来越多的人的重视[43,44]。

模拟退火算法的原理来自物理学,是将优化问题和统计力学的热平衡进行类比,问题的解视作状态,最优解视作退火过程中能量最低的状态,优化的目标函数视作能量函数,模拟物理学中固体物质的退火处理的过程。首先对物体进行加温使之具有足够高的能量,然后再逐步降温,其内部能量也逐渐下降。在热平衡的条件下,物体内部处于不同状态的概率服从 Boltzmann 分布,如果退火的步骤恰当,

那么最后会形成能量最低的状态。该算法在求解优化问题时，不仅接受对目标函数有改进的状态，而且能够以一定的概率接受目标函数恶化的状态，从而可以使得算法避免过早收敛到某个局部最优值，也正因为这种概率性的扰动，使得算法能够跳出局部极值，获得较好的优化结果[45,46,47]。

以最小值优化问题($\min f(x),\ x \in S$)为例，给出基本模拟退火算法的求解流程[6]。

步骤1 随机生成初始解 x_i，设定初始温度 T_0 和终止温度 T_f，令迭代次数 $k=0,\ T_k=T_0$。

步骤2 从 x_i 的邻域 $N(x_i)$ 生成一个解 x_j，并计算两个解对应的目标函数值的增量 $\Delta f = f(x_j) - f(x_i)$。

步骤3 若 $\Delta f < 0$，则令 $x_i = x_j$；否则生成0到1之间的随机数 r，若 $\exp(-\Delta f/T_k) > r$，则令 $x_i = x_j$。

步骤4 若达到热平衡条件(内循环次数大于 $n(T_k)$)，则转步骤5；否则转步骤2。

步骤5 降低 T_k，令 $k=k+1$，若 $T_k < T_f$，则算法停止；否则转步骤2。

目前，模拟退火算法求解的离散优化问题有：背包问题、独立集问题、旅行商问题、排序问题、图着色问题、权匹配问题和选址问题等。此外，算法还可以用于求解连续优化问题。

1.3 遗传算法

遗传算法是一种基于生物进化论中"自然选择、适者生存"规律的优化方法，算法的基本原理源于自然选择理论和遗传机制，通过模拟生命进化的方法来搜索问题的最优解[48,49]。1967年，Holland的学生Bagley在其博士论文中首次提出"遗传算法"一词。1975年，Holland发表了在遗传算法领域具有重要意义的专著——《Adaptation in natural and artificial systems》(《自然系统和人工系统的自适应》)，这是第一本比较系统论述遗传算法的著作。因此，有人把1975年作为遗传算法的诞生年。自提出以来，遗传算法获得了广泛的应用，为许多复杂困难的问题提供了有效的解决办法。

遗传算法在求解优化问题时，采用群体搜索策略而不是在单点上进行寻优，采用随机转换规则而不是确定性原则进行工作。算法将问题的解通过编码的方法

表示成“染色体”，再进行复制、交叉和变异等操作，对“染色体”不断进化，最终收敛到最佳个体，从而获得问题的最优解[50-52]。采用遗传算法求解优化问题时，要对变量进行编码。这样做的原因是在遗传算法中，问题的解是采用字符串来表示的，并且算法也是直接对串进行操作的。常见的编码方法有二进制编码和实数编码。当优化过程结束后，算法再对获得的结果进行解码。复制操作是指从群体中选择优良个体并且淘汰劣质个体。交叉操作是指把父代个体的部分结构进行替换重组从而生成新个体。变异操作是指按概率随机地改变个体的某些基因的取值。

遗传算法在具体应用中有各种变形，但是算法的框架基本类似，主要步骤可以描述为：

步骤 1　问题解的染色体表示。

步骤 2　生成初始群体。

步骤 3　计算各个体的适应度。

步骤 4　基于复制操作选择父代个体。

步骤 5　进行交叉和变异操作，产生新的群体。

步骤 6　判断是否满足停机条件，若不满足，则转步骤 3；否则，算法终止。

目前，遗传算法的应用研究已经渗透到很多领域，并且在许多科学和工程问题中具有良好的应用前景。具体涉及的问题有函数优化、组合优化（如旅行商问题和装箱问题等）、超大规模集成电路的布线、飞机外形设计、管道优化设计、文档自动处理、国民经济预测模型等。

1.4　蚁群优化算法

蚁群优化算法是一种源自大自然生物世界的仿生类算法，算法充分吸收了蚁群在觅食过程中的行为特性。蚁群优化算法最早由 Dorigo 于 1991 年在其博士论文中提出，后期研究工作则由 Dorigo 和其同事共同进行[53,54]。1996 年，Dorigo 等在《IEEE Transactions on Systems, Man, and Cybernetics-Part B》发表了“Ant system: optimization by a colony of cooperating agents”一文，奠定了蚁群优化算法的基础。蚁群优化算法通过其内在的搜索机制，已经成功用于许多组合优化问题的求解中。因为算法仿真中使用的是人工蚂蚁的概念，因此有时也被称为人工蚂蚁系统。

根据昆虫学家长期的观察和研究，发现昆虫世界的蚂蚁能够在没有任何的可

见提示下搜索到从其巢穴到食物源的最短路径，而且能够随着环境的变化而变化，适应性地寻找新的路径，形成新的选择。蚂蚁在搜索食物源的过程中，能够在其走过的路线上释放一种蚂蚁特有的化学物质——信息素，使得一定区域内的其他蚂蚁能感觉到并由此影响它们以后的搜索行为。信息素实际上是蚂蚁之间的通信媒介。当某些路径上经过的蚂蚁越来越多时，其所释放的信息素也越来越多，以致路径上的信息素浓度增大(随着时间的推移信息素会逐渐挥发)，后来蚂蚁选择该路径的可能性也变大，并且会进一步增加该路径上的信息素浓度。随着越来越多的蚂蚁经过该路径，一条最优路径会逐渐形成。蚂蚁这种选择最佳路径的过程称作为蚂蚁的自催化行为。因为其原理是一种正反馈机制，因此，也可以将蚂蚁系统理解为一种增强型学习系统[55—57]。

基本蚁群优化算法的实现步骤可简单描述为：

步骤 1　设置参数，群体初始化。

步骤 2　评价蚁群。

步骤 3　利用状态转移规则生成新的蚁群。

步骤 4　按信息素更新方程修改信息素浓度。

步骤 5　若不满足算法停止条件，则转步骤 2；否则，转步骤 6。

步骤 6　输出当前的最优解。

自蚁群优化算法在经典的旅行商问题和二次分配问题取得成效后，已经逐步渗透到其他优化问题中，例如：图着色问题、车辆调度问题、工件排序问题、通信网络中的负载平衡问题和超大规模集成电路设计等。这种来自自然界的现代启发式优化算法已经在很多方面表现出很好的优化性能，其求解问题的领域也在逐渐扩大。

1.5　微粒群优化算法

微粒群优化算法，或称为粒子群优化算法，是一种基于鸟群觅食行为规律而提出的群体智能优化方法[58, 59]。微粒群优化算法最早是在 1995 年由 Eberhart 和 Kennedy 共同提出的，其基本思想主要是受到他们早期对鸟类群体行为建模和仿真的研究结果的启发。Eberhart 和 Kennedy 在 1995 年的 IEEE International Conference on Neural Networks 和 6th International Symposium on Micromachine and Human Science 上分别发表了题为“Particle swarm optimization”和“A new optimizer

using particle swarm theory"的论文，标志着微粒群优化算法的诞生。由于算法概念简明，易于实现，适合科学研究和工程应用，已经受到越来越多的研究者的重视。

生物群体内个体之间的相互作用产生的群体智能，往往能够给某些问题的求解提供有效的方法。鸟群在寻找食物的过程中，个体之间存在信息交流和共享，每个成员可以从其他成员的搜索经验中获益，并调整自身的搜索行为。当食物源不可预测其分布状态时，这种个体之间的协作所产生的优势是巨大的。基于信息交流和共享的个体间的协作正是微粒群优化算法进行优化搜索的基础[60,61]。在微粒群优化算法中，将优化问题的搜索空间类比作鸟类的飞行空间，并将每只鸟抽象为一个没有质量的微粒，微粒的位置表示问题的候选解，所需要找到的最优解相当于要搜索的食物。微粒群优化算法还为每个微粒设置了类似于鸟类运动行为的简化规则，从而能够使得整个微粒群的运动表现出和鸟类觅食类似的特征，进而可以用于解决优化问题[62]。

基本微粒群优化算法的具体实现步骤如下：

步骤 1 随机初始化群体中各微粒的位置和速度。

步骤 2 评价群体中所有微粒的适应度值。

步骤 3 对每个微粒，将其当前适应值和其个体历史最优位置所对应的适应值进行比较。如果当前的适应值更优，那么利用当前位置更新其历史最优位置。

步骤 4 对每个微粒，将其历史最优适应值和群体内或其邻域内所经过的最好位置的适应值进行比较，若前者更好，则将其作为当前群体内或其邻域内的最优位置。

步骤 5 更新每个微粒的速度和位置。

步骤 6 若没有达到终止条件，则转步骤 2。

目前，微粒群优化算法已经被广泛用于函数优化、神经网络训练、组合优化、模糊系统控制等领域。算法比较有潜力的应用领域包括多目标优化、系统设计、模式识别、车间作业调度、机器人路径规划、时频分析和图像处理等。

第二章　引力搜索算法概述

2.1　基本引力搜索算法

2.1.1　算法的基本思想

万有引力定律是 Newton 于 1687 年在《自然哲学的数学原理》上提出的，万有引力定律是解释物体之间相互作用关系的定律，是物体间由于它们的引力质量而引起的相互吸引力所遵循的规律。万有引力定律的发现，是 17 世纪自然科学最伟大的成果之一。它把地面上物体运动的规律和天体运动的规律统一起来，对物理学和天文学的发展具有极其重要的影响。同时万有引力定律的提出，给人们探索大自然的奥秘建立了极大的信心，在人类认识自然的历史上树立了一座里程碑。自此开始，人们相信，人类有能力理解自然界的各种事物和它们运行的内在规律。

自然界中任何两个物体都是相互吸引的，万有引力普遍存在于任意两个有质量的物体之间。万有引力定律表示如下：自然界中任何两个物体都是相互吸引的，引力的大小跟这两个物体的质量的乘积成正比，跟它们的距离的二次方成反比[65, 66]。数学表达式为：

$$F = G\frac{m_1 m_2}{r^2} \tag{2.1}$$

其中，F 表示两个物体间的引力，G 表示万有引力常数，m_1 和 m_2 分别表示物体 1 和物体 2 的质量，r 表示两个物体间的距离。

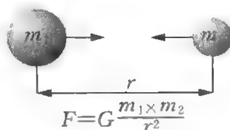


图 2.1　万有引力定律示意图

受万有引力定律启发，Rashedi 等学者在 2009 年提出了一种新型群体智能优化算法——引力搜索算法[63]。作为一种通用型优化算法，它吸收了万有引力定律的特点，通过其内在的搜索机制，已经在一系列困难的优化问题的求解中取得成效。

引力搜索算法在求解优化问题时，搜索个体的位置和问题的解相对应，并且还要考虑个体质量。个体质量用于评价个体的优劣，位置越好，质量越大。由于引力的作用，个体之间相互吸引并且朝着质量较大的个体方向移动，个体运动遵循 Newton 第二定律。随着运动的不断进行，最终整个群体都会聚集在质量最大个体的周围，从而找到质量最大的个体，而质量最大个体占据最优位置。因此，算法可以获得问题的最优解[63]。

引力搜索算法属于群体智能优化算法，而群体智能优化算法最显著的特点是强调个体之间的相互作用。这里，相互作用可以是个体间直接或间接的通信。在引力搜索算法中，万有引力相当于是一种信息传递工具，实现个体间的优化信息共享，整个群体在引力的作用下进行优化搜索。信息的交互过程不仅在群体内部传播了信息，而且群体内所有个体都能处理信息，并根据其所得到的信息改变自身的搜索行为，这样就使得整个群体涌现出一些单个个体所不具备的能力和特性。也就是说，在群体中，个体行为虽然简单，但是个体通过得到的信息相互作用以解决全局目标，信息在整个群体的传播使得问题能够比由单个个体求解更加有效的获得解决。

此外，从引力搜索算法设计的起源来看，是物理原理和优化思想相结合的产物，具体是通过对万有引力定律的模拟来实现的。其实，早在 1953 年，就已存在通过模拟物理学有关原理来构造优化算法的案例，最典型的算法是模拟退火算法，该算法是模拟统计物理中固体物质的结晶过程。因此，引力搜索算法可以视为物理学原理在优化方法中的又一次成功应用。

2.1.2　算法的模型

引力搜索算法首先在解空间和速度空间分别对位置和速度进行初始化，其中位置表示问题的解。例如，D 维空间中的第 i 个搜索个体的位置和速度分别表示为 $\boldsymbol{X}_i=(x_i^1, \cdots, x_i^d, \cdots, x_i^D)$ 和 $\boldsymbol{V}_i=(v_i^1, \cdots, v_i^d, \cdots, v_i^D)$，其中 x_i^d 和 v_i^d 分别表示个体 i 在第 d 维的位置分量和速度分量。通过评价各个个体的目标函数值，

确定每个个体的质量和受到的引力,计算加速度,并更新速度和位置。

(1) 计算质量

个体 i 的质量定义如下:

$$q_i(t)=\frac{fit_i(t)-worst(t)}{best(t)-worst(t)} \tag{2.2}$$

$$M_i(t)=\frac{q_i(t)}{\sum_{j=1}^{N} q_j(t)} \tag{2.3}$$

其中,$fit_i(t)$和 $M_i(t)$分别表示在第 t 次迭代时第 i 个个体的适应度函数值和质量;$best(t)$和 $worst(t)$表示在第 t 次迭代时所有个体中最优适应度函数值和最差适应度函数值,对最小化问题,其定义如下:

$$best(t)=\min_{j\in\{1,2,\cdots,N\}} fit_j(t) \tag{2.4}$$

$$worst(t)=\max_{j\in\{1,2,\cdots,N\}} fit_j(t) \tag{2.5}$$

(2) 计算引力

算法源于对万有引力定律的模拟,但不拘泥于物理学中万有引力公式的精确表达式。根据实验结果,采用下述的引力定义表达式效果更好。在第 d 维上,个体 j 对个体 i 的引力定义如下:

$$F_{ij}^d(t)=G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t)-x_i^d(t)) \tag{2.6}$$

其中,$G(t)$表示在第 t 次迭代时万有引力常数的取值,$G(t)=G_0e^{-\alpha t/T}$,G_0 和 α 为常数,T 表示最大迭代次数;$R_{ij}(t)$表示个体 i 和 j 之间欧氏距离,$i,j\in\{1,2,\cdots,N\}$,且 $i\neq j$;$d=1,2,\cdots,D$;ε 是一常数,防止分母为零。

在第 d 维上,个体 i 所受的合力为:

$$F_i^d(t)=\sum_{j\in kbest,\ j\neq i}^{N} rand_j F_{ij}^d(t) \tag{2.7}$$

其中,$rand_j$ 表示在[0, 1]之间服从均匀分布的一个随机变量;$kbest$ 表示个体质量按降序排在前 k 个的个体,并且 k 的取值随迭代次数线性减小,初值为 N,终值为 1。

(3) 计算加速度

根据 Newton 第二定律，个体 i 在第 d 维的加速度方程为：

$$a_i^d(t)=\frac{F_i^d(t)}{M_i(t)} \tag{2.8}$$

(4) 更新速度和位置

$$v_i^d(t+1)=r\times v_i^d(t)+a_i^d(t) \tag{2.9}$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1) \tag{2.10}$$

其中，r 表示在[0, 1]之间服从均匀分布的一个随机变量。

目前，已有的智能计算理论和应用研究表明智能优化方法是一种能够有效解决大多数优化问题的方法。基于智能特征的优化算法设计必须遵守简单有效的原则，对于自然现象过于复杂的模拟往往会导致算法不具有推广性和实用价值，许多智能优化算法不成功的原因就在于此。如何从自然规律中提炼出有效的规则应用到算法的寻优策略，是智能优化算法设计时要面对的一个关键问题。引力搜索算法的目的并不是为了拘泥地模拟万有引力定律，而是利用万有引力定律的特点去解决优化问题。算法受万有引力定律启发，但是不拘泥于万有引力公式的原始表达式。根据实验结果，在算法中引力与两个个体质量的乘积成正比和它们的距离成反比的优化效果更好。此外，同样根据实验情况，万有引力常数不再固定不变，而是随迭代次数单调递减，算法的优化性能更好。

在计算个体受到的万有引力的合力时，算法只考虑质量较大的个体产生的引力。因为在引力搜索算法中，当引力较大时，或者有质量较大的个体，或者两个个体间的距离较小。质量较大的个体占据较优的位置，并且代表较好的解。算法仅考虑来自质量较大的个体的引力，可以消除因距离较小而引力较大的影响，引导其他个体向质量较大的个体方向移动。在引力的不断作用下，整个群体逐渐向质量最大的个体方向逼近，最终搜索到问题的最优解。

2.1.3 算法的流程

基本引力搜索算法主要包含三个组成部分：群体初始化、计算个体质量和所受的引力以及个体运动更新。算法的主要实现步骤如下：

步骤 1 随机初始化群体中各个体的位置，个体的初始速度为零。

步骤 2　计算每个个体的适应度函数值。

步骤 3　计算个体的质量和受到的引力。

步骤 4　计算个体的加速度和速度。

步骤 5　更新个体的位置。

步骤 6　若满足终止条件,则输出当前结果并终止算法,否则转向步骤 2。

上述程序的结构流程如图 2.2 所示。

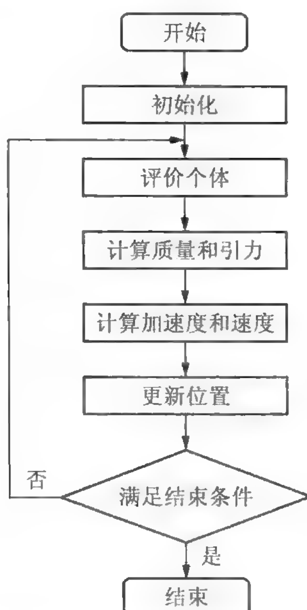


图 2.2　引力搜索算法的流程图

引力搜索算法是一种新兴的智能优化算法,其基于万有引力定律的搜索机制和现有智能优化算法的搜索机制有着本质的区别。如,遗传算法的基本思想源于生物学中的自然选择原理和遗传机制;蚁群优化算法是根据蚂蚁觅食原理而设计出的;微粒群优化算法是模拟鸟群飞行觅食的行为。这里,对引力搜索算法的特点做如下总结:

(1) 每个搜索个体都赋予 4 个状态变量,分别为位置、速度、加速度和质量。位置用于表示问题的解,速度用于更新位置,加速度用于更新速度,质量用于评价

个体的优劣。

（2）整个群体总是寻找质量最大的个体，无论是最大值优化问题还是最小值优化问题，都可以通过质量函数的定义，将优化目标转换为搜索质量最大的个体。

（3）从引力搜索算法设计的起源来看，算法主要是对万有引力定律的模拟，是将物理原理和优化思想相结合而产生的。引力搜索算法最显著的特点是整个群体依靠个体之间的引力作用进行寻优，引力相当于一种优化信息的传递工具。根据算法的设计，个体的质量越大，引力也越大。因此，在引力作用下，整个群体能够向质量最大的个体方向移动，从而能够搜索到问题的最优解。

（4）引力搜索算法的流程简单，参数设置少，可以很好的和各种优化问题相结合，易于实现。

除了上述这些特点之外，引力搜索算法也具有智能优化算法一些共同的特点。例如，引力搜索算法对目标函数没有特别要求，不要求函数具有连续和可导等数学性质，甚至有时连有没有解析表达式都不做要求，而且对问题中不确定的信息具有一定的适应能力，因此，算法的通用性比较强。此外，从算法实现的方法来看，引力搜索算法可以采用串行或并行的方法实现，可以根据具体问题，设计出合理的算法实现方式。

2.2　引力搜索算法的系统学特征

引力搜索算法作为一种新的求解复杂优化问题的方法，在其优化机理中，体现出系统科学的方法论。分析引力搜索算法中所蕴含的系统科学思想，有助于我们更好地利用该算法来处理所遇到的问题。

2.2.1　系统性

系统科学的中心概念是系统，其基本的特点是强调整体性。从系统自身的规定性来看，按照现代系统论的创立者 Bertalanffy 所给出的定义，系统是“相互作用的多元素的复合体”。把该定义稍加精确化，可以进一步表述为：如果一个集合中至少有两个可以区分的对象，并且所有对象按照可以辨认的特有方式相互联系在一起，就称该集合为系统。系统的基本特点是：(1)多元性，系统中存在有差别的多个元素，组分的差异性和多样性是系统“生命力”的重要源泉；(2)相关性或相干性，系统不存在与其他元素无关的组分或孤立元素，差异而不相干的元素形不成系统，

相关性同样是系统“生命力”的重要源泉；(3)整体性，系统是它所有元素构成的一个整体[67, 68]。

将引力搜索算法本身视为一个整体，就会发现它具备了系统的三个基本特征。算法中的群体是由一定数量的互异个体组成，体现出系统的多元性；个体行为的相互影响体现出系统的相关性；在算法中，采用群体搜索的求解结果要明显好于单个个体的结果，整个群体可以完成个体所完成不了的任务，体现出系统的整体性，体现出整体大于部分之和的整体突现原理。

2.2.2 分布式

在引力搜索算法中，群体行为体现出分布式的特征。当整个群体要完成一项工作时，每个个体都在为实现这一目标而进行工作，最终任务的完成不会因为某个或某些个体的缺陷而受到影响[67, 68]。

对优化问题而言，从一点出发的单点搜索往往会受到某些局部特征的限制，可能得不到问题的最优解或满意解；而引力搜索算法可视为一个分布式多智能体的系统，在所求解问题空间的多个点同时进行搜索，不仅使算法具有较好的全局优化能力，而且最终问题的求解不会因为某个或某些个体无法成功获得解而受到影响。

2.2.3 自组织性和涌现性

自组织是系统科学的一个重要概念，是系统演化时出现的一种现象。在系统实现空间的、时间的或功能的结构过程中，如果没有外界的特定干扰，仅仅是依靠系统内部的相互作用来实现的，便称该系统是自组织的。引力搜索算法是一种自组织算法，优化过程体现出从无序到有序的演化过程。在引力的作用下，个体之间相互影响并实现优化信息的共享。群体从无序的随机搜索逐渐向最优解逼近，体现系统的自组织演化。

在自组织系统中，只需要设定个体行为的“局部规则”，无须规定群体行为的“全局规则”，通过个体间的交互作用，就能产生出整体具有而部分或部分总和所没有的东西，如整体的特性，整体的行为，整体的功能等。一旦将系统分解成它的组成部分，这些东西便不复存在。系统科学将这种整体才具有、孤立的部分以及其总和不具有的特性，称为涌现。涌现性通俗的表达，就是“整体大于部分之和”。涌现

是由系统的组成成分相互作用而激发出来的，是一种组成成分之间的相干效应。个体的相互作用才是整体的基础。在算法中，通过引力的作用，个体之间相互影响，并且个体行为表现出主动性或适应性的特征。个体能够根据得到的优化信息的不同，而对自身的行为进行不同的调整。每次迭代并不是个体间的相互取代，而是个体知识或经验的更新，尽管没有中心控制，但整个群体却表现出强大的解决复杂问题的能力[68,69]。

2.2.4 反馈机制

系统学观点认为：反馈是把系统现有行为和现有行为的结果作为影响未来行为的因素。一般情况下，反馈可以分为正反馈和负反馈两种。以现有的行为来加强未来的行为，是正反馈，以现有的行为来削弱将来的行为，是负反馈。

在引力搜索算法中，每个个体构造的解存在差异，较优的个体质量较大，其引力也较大，并且由此能够吸引更多的个体。这个正反馈的过程可以引导群体向最优解方向进化。正反馈是引力搜索算法的重要特征，保证算法的演化过程能够进行。引力搜索算法中也隐含着负反馈机制，通过构造问题的解时，采用引入随机变量的方法实现的。这种方法增强了生成新解的随机性，一方面可能会产生在一定程度上退化的解，但另一方面又使得个体的搜索范围保持足够的大[57]。这样，在引力搜索算法中，正反馈机制缩小搜索区域，引导群体向最优解方向进化；负反馈机制保持一定的搜索范围，防止算法早熟收敛。在正反馈和负反馈的共同作用和影响下，算法得以自组织进化，最终获得问题的最优解或满意解。

2.3 引力搜索算法的研究进展

2.3.1 引力搜索算法的提出和改进

通过对万有引力定律的模拟，Rashedi 等人于 2009 年提出了引力搜索算法。算法以典型的单目标连续优化测试函数为基准，并与遗传算法和微粒群优化算法等经典智能优化算法进行比较。实验结果表明引力搜索算法具有较好的优化性能。随后，在最初算法是实数编码的基础上，Rashedi 等人在 2010 年又提出一种二进制编码的引力搜索算法[64]，算法在与二进制编码的遗传算法和微粒群优化算法的比较中取得较好的结果。自此，引力搜索算法开始受到越来越多的研究者的关注。但是任何事物都具有两面性，引力搜索算法作为一种新型的优化方法也不例

外。算法一方面具有良好的优化潜力，另一方面存在局部搜索能力较差和易早熟收敛等问题。随着对这一算法的理解不断加深，不少研究人员开始对这些存在的问题进行分析，并提出了一些改进方法。

(1) 增加扰动操作

受天体物理学启发，文献[70]提出一种扰动操作。在算法每次迭代时，计算每个个体和其最近个体的距离以及该个体和当前最优个体的距离，并计算两者的比值。如果该比值小于设定的阈值，就对该个体进行扰动。在算法的优化过程中，阈值随迭代次数逐渐减小。这一设置能够使得算法在迭代早期具有较强的全局探索能力而在后期具有较强的局部开发能力。实验结果表明新算法的优化精度有所改善。

(2) 修改质量函数定义

质量函数的设计是引力搜索算法的一个关键因素，个体质量的大小体现出对应解的优劣。在基本引力搜索算法中，质量函数根据个体的适应度函数值进行计算。文献[71]提出一种基于权值的引力搜索算法，在每次的迭代过程中，给每个个体赋予一个权值。权值的作用使得质量大的个体的质量更大，质量小的个体的质量更小。实验结果表明，与基本引力搜索算法相比，新算法的优化性能有所提高。

(3) 使用三角范数算子

文献[72]采用三角范数中的算子来替换引力计算公式中的乘法算子，并由此构造一个模糊集。算法的设计保证个体所受到的引力大小与三角范数算子的函数值构成正比关系。而在三角范数的各个算子中，越强的算子，其函数值也越大；越弱的算子，其函数值也越小。因此，不同三角范数算子对算法优化性能的影响程度有差异，通过实验比较发现，部分三角范数算子可以改进基本引力搜索算法的优化性能。此外，文献[72]还对引入三角范数算子后可能出现个体质量为零的情况进行了分析，并给出解决方案。

(4) 与其他算法的融合

利用不同算法的优化行为、优化结构和优化机制的互补性来提高算法的优化性能，将多种算法进行混合已经成为优化算法发展的一种重要方法。因此，发展和其他算法融合的引力搜索算法是改进引力搜索算法的一条有效途径。文献[73]将微粒群算法和引力搜索算法相结合，给出一种混合型的引力搜索算法。两种算法的融合主要是将微粒群算法中的群体经验借鉴到引力搜索算法中，提出一种新的

速度更新方程，仿真结果显示了新算法求解优化问题的效果良好。文献[74]对差分进化算法和引力搜索算法进行了分析和比较，并将差分进化算法中的变异、交叉和选择操作引入到引力搜索算法中，实验结果表明新算法可有效提高基本引力搜索算法的局部优化能力。

（5）多目标优化

文献[75]第一次将引力搜索算法用于求解多目标优化问题。在该算法中，移动个体的质量设为单位 1，而存档个体的质量根据在目标空间中该个体与其最近邻居的距离进行计算。这种策略能够使得存档个体尽可能的均匀分布，类似于小生境技术中的适应度共享方法。移动个体在存档个体的引力作用下，朝 Pareto 最优解方向进化。之后，文献[76]也提出一种多目标引力搜索算法。这两种算法的实验结果都表明引力搜索算法能够用于求解多目标优化问题，但优化性能有待进一步提高。

2.3.2 引力搜索算法的应用

鉴于引力搜索算法的有效性和通用性，算法在一些领域中得到成功应用。可简单归纳为如下几个方面：

（1）聚类分析

聚类分析的研究始于 20 世纪 60 年代，是一个经典的优化问题。聚类是对数据集进行分类，使得类内相似性最大，而类间相似性最小。文献[77]提出了一种求解聚类分析的引力搜索算法，首先对寻优个体随机初始化，然后在引力作用下对问题空间进行搜索；文献[78]结合 K 调和均值的方法，给出一种求解聚类问题的混合引力搜索算法；文献[79]在求解聚类问题时，先利用引力搜索算法进行全局搜索，再采用一种启发式策略进行局部优化。

（2）PID 参数优化

PID(Proportion Integration Differentiation)控制器算法实质是对“过去”、“现在”和“将来”的信息进行估计的方法。据统计，目前在控制工程领域仍然有约 90%的控制回路具有 PID 结构。而任何 PID 控制器性能的优劣完全取决于其控制参数的设置。文献[80]将引力搜索算法用于直流电动机的 PID 控制器的参数优化，算法以估计最小均方误差为目标函数。实验结果表明，与传统算法 Ziegler-Nichols 相比，该算法在上升时间、调节时间和超调量等指标上表现出较好的优化结果。

(3) 经济负荷分配

经济负荷分配是电力系统规划和运行中的一类典型的优化问题，其目标是要求在满足电力需求与电机特征等约束条件下，如何在各台发电机组之间分配负荷使得总发电成本最少。最小化发电成本对提高电力系统运营的经济性和可靠性有着重要意义。文献[81]结合罚函数法，利用引力搜索算法有效求解了一类带有等式约束的经济负荷分配问题。文献[82]将OBL(Opposition-Based Learning)技术和引力搜索算法相结合，提出了一种混合型的求解经济负荷分配的引力搜索算法，在求解多个典型测试问题时表现出良好的优化性能。

(4) 特征选择

特征选择是统计学、模式识别和机器学习等领域的一个研究热点问题。其目的是要从大规模的样本空间中挖掘出隐藏的有意义的特征数据，并以此分析和研究事物内在的规律。文献[83]结合OPF(Optimum-Path Forest)方法，给出一种基于引力搜索算法的特征选择方法。在该算法中，利用引力搜索算法求出问题的解，并利用OPF方法对解所对应的特征子集进行训练。在对多个特征选择的数据测试和与微粒群优化算法的比较中，引力搜索算法都取得比较满意的结果。

(5) DNA编码序列设计

从DNA计算诞生以来，DNA编码序列设计就一直是该领域的一个研究热点问题。DNA编码序列设计的目的是要通过序列的优化尽可能地减少在NDA计算过程中的错误杂交。NDA计算的精确度与可靠性和DNA编码序列设计的好坏紧密相关。文献[84]将引力搜索算法应用到DNA编码序列设计的研究中，结合编码问题的特征，给出了质量函数的定义。实验结果表明，与其他一些智能优化算法相比，引力搜索算法的求解结果更好。

(6) 流水线调度问题

流水线调度问题是一类广泛应用的调度问题，有着重要的理论价值和实践意义。目前，越来越多的研究者开始关注这个问题。流水线调度是在一定的约束条件下，研究在n台机器上的m个工件的流水加工过程。文献[85]将引力搜索算法用于优化流水线调度问题，并采用了最大排序规则、边界变异和局部搜索等方法。仿真实验结果表明引力搜索算法优于遗传算法和微粒群优化算法等算法。

(7) 其他问题

除了以上的应用外，引力搜索算法还被成功用于图像处理[86]、网络服务选择

问题[87]、系统辨识[88]、天线阵列综合[89]、关联规则挖掘[90]、石油需求量的预测[91]和滤波器建模[92]等。

2.3.3 引力搜索算法的研究展望

引力搜索算法有望成为继遗传算法、蚁群优化算法和微粒群优化算法等算法之后又一个优秀的智能优化算法，将会得到更多的国内外研究者的关注。但算法自提出以来，至今也只有短短的几年发展时间。和上述这几个成熟的算法比较而言，引力搜索算法还很年轻，还有很多的工作需要进一步研究和探讨：

(1) 算法的理论研究

虽然引力搜索算法在求解很多问题时表现出良好的优化效果，但是其相关的理论研究非常稀缺。算法的机理缺少深刻并且具有普遍意义的系统分析和探讨，同时算法的数学基础还很薄弱，目前还没有见到有关算法收敛性的证明。因此，需要对引力搜索算法的机理、收敛性、收敛速度等理论进行系统研究，阐明算法的工作原理和性态，为算法的发展和应用提供相应的理论依据。

(2) 算法的改进

注重高效的引力搜索算法的开发，例如对算法核心的迭代方程(包括速度和位置的更新方程)进行改进，给出算法参数自适应调整策略等。另外，从解决问题的角度出发，本着优势互补的原则，充分发挥各种算法的优点，混合算法也是解决问题的一种发展趋势。应深入分析引力搜索算法和其他算法(包括传统优化方法和智能优化算法)在优化原理、搜索模式和优化能力等方面的互补性，在此基础上，设计出基于引力搜索算法的混合型算法或融入新型优化思想的引力搜索算法。

(3) 算法的应用

目前，引力搜索算法的应用研究还处于起步阶段，所求解的问题相对有限，实际应用还未挖掘出其真正的潜力。基于算法良好的优化性能，其应用前景将十分广阔。一方面应拓宽和深化算法的应用领域，将其更广泛更深入地用于电力系统、机械设计、自动控制、通信网络和生物信息等领域；另一方面将算法用于求解多目标优化问题。在现实问题中，存在大量的多目标优化问题，而这些问题具有广泛性、代表性和重要性，研究和推广基于引力搜索算法的多目标求解方法具有重要的意义。引力搜索算法可以为许多优化问题的求解提供新方法，而这些应用研究对算法的理解和改进具有重要意义。此外，已经有学者开始尝试将引力搜索算法用

于求解离散优化问题，将算法推广到离散优化领域，为求解离散问题提供新方法。但是如何设计有效的操作算子，需要进一步分析和研究。

引力搜索算法是一种新兴的智能优化算法，目前算法还有很多未完善的方面，需要对其进行深入分析和研究。但是可以推断的是，随着研究的不断深入，引力搜索算法必将展现出更加广阔和更加光明的发展前景。

第三章　单目标连续优化的引力搜索算法

单目标连续优化问题是优化理论中的一个重要组成部分，是许多优化问题的研究基础，也是许多科学和工程领域中需要求解的问题。因此，对该问题的研究有着重要的理论价值和实际意义。不失一般性，以最小值优化问题为例，给出其数学模型：

$$\min f(x),\ x \in S \tag{3.1}$$

其中，$f(x)$为目标函数；$S \subset R^D$ 为搜索空间，也可以称为可行域；$x = (x_1, x_2, \cdots, x_D)$ 是一个 D 维决策变量，且 $x \in S$。通常情况下，每个自变量 x_i 要满足一定的边界约束限制：

$$l_i \leqslant x_i \leqslant u_i \tag{3.2}$$

这里，l_i 和 u_i 分别表示第 i 个变量的下界和上界，$i = 1, 2, \cdots, D$。

单目标优化问题的数学描述虽然简单，但是当问题的目标函数拓扑结构比较复杂，决策变量的数目比较多时，其求解的难度会显著增加。同时人们往往无法获得有关问题的任何先验知识，没有具体规律可循，也会给问题的求解带来不便。此外，随着时代的发展，我们所要解决的问题往往是不连续、不可微、非线性、非凸的、多极值的，这又进一步增加了问题的求解难度。采用传统优化算法（如最速下降法和共轭梯度法等）很难获得理想的优化效果。近几十年来，智能优化算法应运而生。这些智能优化算法的搜索原理完全异于传统优化算法，为优化问题的求解提供了新的思路和途径。实践证明这些智能优化算法在很多优化问题的应用中能够获得成功。本章采用引力搜索算法求解单目标连续优化问题，分析算法的收敛性，并提出一种改进的引力搜索算法。从理论和实验两个角度对算法展开研究，充分验证算法的可行性和有效性。

3.1 收敛性分析

采用引力搜索算法求解函数优化的基本思想已经在 2.1 节中介绍过，算法主要包括计算个体质量、所受到的引力和加速度，更新速度与位置等操作。从现有的研究成果来看，引力搜索算法已经呈现出可以大范围应用的可能性，然而其理论基础尚很薄弱。引力搜索算法是一种新型智能优化算法，其创立至今不超过 5 年时间，理论成果非常稀缺，缺少算法收敛性的分析。

从引力搜索算法的定义和流程可以看出，随机变量的存在使得算法具有随机性，而求解过程同时具有迭代性。因此，实际上引力搜索算法是一种随机迭代算法，采用随机压缩映射定理，给出算法的收敛性证明。

在本章中，若无特别说明，(Ω, A, p)表示完全概率测度空间。

定义 3.1[93, 94] 设 X 是非空集合，对于 X 中的任意两个元素 x 和 y，按照某一法则都对应唯一的实数 $d(x, y)$，且满足下述三条公理：

(1) 非负性：$d(x, y) \geqslant 0$；$d(x, y) = 0$ 当且仅当 $x = y$；

(2) 对称性：$d(x, y) = d(y, x)$；

(3) 三角不等式：对于任意的 $x, y, z \in X$，都有 $d(x, y) \leqslant d(x, z) + d(z, y)$。

则称 d 为 X 上的度量，称(X, d)为度量空间。

定义 3.2[93, 94] 设有映射 $T: \Omega \times X \to X$，$\forall x \in X$ 取定，令

$$T(\omega)x = y(\omega) \tag{3.3}$$

若 $y(\omega)$为一 X—值随机变量，则称 T 为一随机算子（随机映射）。

定义 3.3[93, 94] 设有映射 $T: \Omega \times X \to X$ 为一随机算子，若 $\xi(\omega)$为 X 一值随机变量，并且满足：

$$T(\omega)\xi(\omega) = \xi(\omega),\ a.s. \tag{3.4}$$

则称 $\xi(\omega)$为随机算子 T 的随机不动点。

定义 3.4[93, 94] 随机算子 $T: \Omega \times X \to X$ 称为随机压缩算子（随机压缩映射），如果存在非负实值随机变量 $k(\omega) < 1$，$a.s.$，使得

$$p(\{\omega: d(T(\omega)x, T(\omega)y) \leqslant k(\omega)d(x, y)\}) = 1 \quad \forall x, y \in X \tag{3.5}$$

定义 3.5[93, 94] 度量空间(X, d)中的点列$\{x_n\}$称为 Cauchy 列，是指对任意的

$\varepsilon > 0$，存在自然数 N，当 $n, m \geqslant N$ 时，有 $d(x_n, x_m) < \varepsilon$。度量空间 (X, d) 是完备的，是指其中的任何 Cauchy 列都是收敛的。

定理 3.1[93, 94] 设随机算子 $T: \Omega \times X \to X$ 满足：对于几乎所有的 $\omega \in \Omega$，$T(\omega)$ 均为随机压缩算子，即 $\exists \Omega_0 \subset \Omega$，$p(\Omega_0) = 1$，使 $\omega \in \Omega_0$，有：

$$d(T(\omega)x, T(\omega)y) \leqslant k(\omega)d(x, y) \quad \forall x, y \in X \tag{3.6}$$

其中 (X, d) 是完备度量空间，$0 \leqslant k(\omega) < 1$，$\omega \in \Omega_0$，则称 $T(\omega)$ 有唯一随机不动点 $\xi(\omega)$，上述定理称为随机压缩映射定理。

以最小值优化问题为例，给出引力搜索算法的收敛性分析。

假设算法的群体规模为 N，P^k 表示第 k 次迭代时的群体，且 $P^k = \{x_1^k, \cdots, x_i^k, \cdots x_N^k\}$，其中 x_i^k 表示第 k 次迭代时第 i 个个体的取值。将所有可能的群体 P^k 构成的集合记为 S。引力搜索算法的迭代方程主要是速度和位置的更新方程。由于随机因素的存在，一次迭代相当于一次随机映射 $T: \Omega \times S \to S$，且该映射 T 可看作由速度和位置两个映射合成。因此，引力搜索算法的迭代过程可以通过随机映射 T 定义为

$$T(\omega)P^{k-1} = P^k \tag{3.7}$$

下面，给出一个对群体 P^k 进行评价的函数 $F(P^k)$ 的表达式：

$$F(P^k) = \frac{1}{N}\sum_{i=1}^{N} f(x_i^k) \tag{3.8}$$

其中，$f(x_i^k)$ 表示 x_i^k 的目标函数值。

定义映射 $d: S \times S \to R$，其中 d 的表达式为

$$d(P^j, P^k) = \begin{cases} |F(P^j) - M| + |F(P^k) - M| & P^j \neq P^k \\ 0 & P^j = P^k \end{cases} \tag{3.9}$$

其中，M 为目标函数值的下界，对于最小值优化问题 M 肯定存在，$P^j, P^k \in S$。

对映射 d 进一步分析，对任意的 $P^j, P^k \in S$，可以得到以下结论

(1) $d(P^j, P^k) \geqslant 0$；$d(P^j, P^k) = 0$ 当且仅当 $P^j = P^k$

(2) $d(P^j, P^k) = d(P^k, P^j)$

(3) $d(P^j, P^k) = |F(P^j) - M| + |F(P^k) - M|$

$$\leqslant |F(P^j) - M| + |F(P^l) - M| + |F(P^l) - M| + |F(P^k) - M|$$

$$= d(P^j, P^l) + d(P^l, P^k)$$

因此，d 为 S 上的度量，(S, d)为度量空间。

利用计算机做数值计算时要受到编码精度的限制，如采用浮点编码时，若编码精度为 δ，则 x_i^k 所有可能的取值情况有 $\frac{b_i - a_i}{\delta}$ 种，P^k 所有可能的取值情况有 $\prod_{i=1}^{N} \frac{b_i - a_i}{\delta}$ 种[95]，这就意味着集合 S 是有限集。这里，b_i 和 a_i 分别表示 x_i^k 的上下限。同时根据文献[96]知，若 S 为有限集，则(S, d)中的任意 Cauchy 列都是收敛的。因此，(S, d)是完备的度量空间。

根据引力搜索算法的设计，在每次迭代时，整个群体在引力的作用下都要向质量较大的个体方向移动。这个移动过程也是个体从较差解区域向较好解区域的逼近过程，个体所对应解的质量会随之改善，相应的目标函数值也会有所降低。从整个优化过程来看，就整个群体而言，$F(P^k)$的函数值随迭代次数的变化会形成一个单调非增序列，即有

$$F(P^k) = F(T(\omega)P^{k-1}) \leqslant F(P^{k-1}) \tag{3.10}$$

在此基础上，可以得到以下结论

$$\begin{aligned} d(T(\omega)P^j, T(\omega)P^k) &= |F(T(\omega)P^j) - M| + |F(T(\omega)P^k) - M| \\ &= |F(P^{j+1}) - M| + |F(P^{k+1}) - M| \\ &\leqslant |F(P^j) - M| + |F(P^k) - M| \\ &= d(P^j, P^k) \end{aligned} \tag{3.11}$$

所以，对映射 T 存在一个非负实值随机变量 $0 \leqslant k(\omega) < 1$, $a.s.$，对任意的 P^j, $P^k \in S$ 有

$$d(T(\omega)P^j, T(\omega)P^k) = k(\omega)d(P^j, P^k) \tag{3.12}$$

这就意味着引力搜索算法迭代所形成的映射是随机压缩映射。

令

$$\Omega_0 = \{\omega: d(T(\omega)P^j, T(\omega)P^k) = k(\omega)d(P^j, P^k)\} \subset \Omega \tag{3.13}$$

则有

$$p(\Omega_0) = 1 \tag{3.14}$$

根据上面的分析，引力搜索算法的迭代过程满足随机压缩映射定理的条件，

算法在无限次迭代后能够保证收敛。若优化问题的目标函数有多个最优解，则可将原有的度量空间划分成多个小的度量空间，保证算法在每个小的度量空间上满足随机压缩映射定理的条件即可。

3.2 改进的引力搜索算法

基于随机压缩映射定理证明了基本引力搜索算法在无限次迭代后能够收敛到最优解，为算法的应用奠定了理论基础。但是在实际应用中通常要求在有限的迭代次数内，算法具有较高的优化精度和较快的收敛速度。而现有的研究成果表明，在有限的迭代次数内，基本引力搜索算法存在求解精度不高和收敛速度慢等问题。因此，需要对基本引力搜索算法进行改进，以提高算法的优化性能，为优化问题的求解提供一种又好又快的方法。

在现有的智能优化算法中，虽然微粒群优化算法和引力搜索算法的优化原理有着本质的区别，但是这两种算法都采用了基于速度和位置的计算模型。在基本微粒群优化算法中，第 i 个微粒在第 d 维上的速度 v_i^d 和位置 x_i^d 的更新方程为

$$v_i^d(t+1) = v_i^d(t) + c_1 r_1 (pbest_i^d(t) - x_i^d(t)) + c_2 r_2 (gbest^d(t) - x_i^d(t)) \tag{3.15}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{3.16}$$

其中，c_1 和 c_2 为学习因子；r_1 和 r_2 为[0, 1]之间的随机数；$pbest_i$ 表示第 i 个微粒迄今为止搜索到的最好位置；$gbest^d$ 表示整个群体迄今为止搜索到的最好位置；$i = 1, 2, \cdots, N$，N 为群体规模；$d = 1, 2, \cdots, D$，D 表示问题的维数。

自微粒群优化算法提出以来，许多学者对算法进行了研究并提出了很多改进的算法。其中，有两种算法最具有代表性。一个是文献[59]提出的带权重的微粒群优化算法，另一个文献[97]提出的是带收缩因子的微粒群优化算法，这两种改进算法极大地提高了基本微粒群优化算法的优化性能，对算法的发展产生了深刻的影响。

在带权重的微粒群优化算法中，一个惯性权重因子 ω 被添加到速度更新方程中，即

$$v_i^d(t+1) = \omega v_i^d(t) + c_1 r_1 (pbest_i^d(t) - x_i^d(t)) + c_2 r_2 (gbest^d(t) - x_i^d(t)) \tag{3.17}$$

在带收缩因子的微粒群优化算法中，一个收缩因子 χ 被引入到速度更新公式中,即

$$v_i^d(t+1)=\chi(v_i^d(t)+c_1r_1(pbest_i^d(t)-x_i^d(t))+c_2r_2(gbest^d(t)-x_i^d(t)))\tag{3.18}$$

将基本微粒群优化算法的速度更新方程带入位置更新方程,可以得到

$$\begin{aligned}x_i^d(t+1)&=x_i^d(t)+v_i^d(t+1)\\&=x_i^d(t)+v_i^d(t)+c_1r_1(pbest_i^d(t)-x_i^d(t))+c_2r_2(gbest^d(t)-x_i^d(t))\\&=(1-c_1r_1-c_2r_2)x_i^d(t)+v_i^d(t)+c_1r_1pbest_i^d(t)+c_2r_2gbest^d(t)\end{aligned}\tag{3.19}$$

从上述位置更新方程(3.19)可以看出,$x_i^d(t)$这一项有系数。同样地,分别将带权重的和带收缩因子的微粒群优化算法的速度更新公式带入位置更新公式,可以得到

$$\begin{aligned}x_i^d(t+1)=&(1-c_1r_1-c_2r_2)x_i^d(t)+\omega v_i^d(t)\\&+c_1r_1pbest_i^d(t)+c_2r_2gbest^d(t)\end{aligned}\tag{3.20}$$

以及

$$\begin{aligned}x_i^d(t+1)=&(1-\chi c_1r_1-\chi c_2r_2)x_i^d(t)+\chi v_i^d(t)\\&+\chi c_1r_1pbest_i^d(t)+\chi c_2r_2gbest^d(t)\end{aligned}\tag{3.21}$$

从这两个位置更新公式(3.20)和(3.21)可以发现,$x_i^d(t)$这一项也有系数。从微粒群优化算法的现有研究成果来看,这些系数对算法优化性能的改善有着一定的影响。

采用同样的方法,将引力搜索算法的速度更新方程带入位置更新方程,得到

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1)=x_i^d(t)+rv_i^d(t)+a_i^d(t)\tag{3.22}$$

该位置更新方程中$x_i^d(t)$不含有系数,受微粒群优化算法的启发,在$x_i^d(t)$前面添加一个系数 s,位置更新方程变为

$$x_i^d(t+1)=sx_i^d(t)+v_i^d(t+1)=sx_i^d(t)+rv_i^d(t)+a_i^d(t)\tag{3.23}$$

其中,s 可以是一个非负的常数,也可以是一个函数,将通过实验确定。

采用随机压缩映射定理,同样可以证明改进后的引力搜索算法的收敛性。下面将通过实验重点分析系数 s 的引入对算法优化性能的影响。

3.3 数值实验

实验共分两个部分展开,首先对系数 s 的取值通过实验确定,然后将改进的引力搜索算法和现有智能优化算法的求解性能进行比较。算法采用 Matlab7.1 语言编程实现,在 Windows XP 操作系统,CPU 为 P4 2.4 GHz 和内存为 1 G 的计算机上运行。

3.3.1 测试函数

采用智能优化算法领域中典型的测试函数进行数值实验。这些函数总共有 13 个,可以从多个角度测试算法的优化性能。这些函数的具体定义如下[98]:

(1) *Sphere Model*:

$$f_1(x) = \sum_{i=1}^{D} x_i^2$$

$-100 \leqslant x_i \leqslant 100$, $\min(f_1) = f_1(0, \cdots, 0) = 0$

(2) *Schwefel's Problem 2.22*:

$$f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|$$

$-10 \leqslant x_i \leqslant 10$, $\min(f_2) = f_2(0, 0, \cdots, 0) = 0$

(3) *Schwefel's Problem 1.2*:

$$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$$

$-100 \leqslant x_i \leqslant 100$, $\min(f_3) = f_3(0, 0, \cdots, 0) = 0$

(4) *Schwefel's Problem 2.21*:

$$f_4(x) = \max_i \{|x_i|, 1 \leqslant i \leqslant D\},$$

$-100 \leqslant x_i \leqslant 100$, $\min(f_4) = f_4(0, 0, \cdots, 0) = 0$

(5) *Generalized Rosenbrock function*:

$$f_5(x) = \sum_{i=1}^{D-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$$

$-30 \leqslant x_i \leqslant 30$, $\min(f_5) = f_5(1, \cdots, 1) = 0$

(6) *Step Function*:

$$f_6(x) = \sum_{i=1}^{D}(\lfloor x_i + 0.5 \rfloor)^2$$

$$-100 \leqslant x_i \leqslant 100,\ \min(f_6) = f_6(0, 0, \cdots, 0) = 0$$

(7) *Quartic Function i.e. Noise*:

$$f_7(x) = \sum_{i=1}^{D} i x_i^4 + rand[0, 1)$$

$$-1.28 \leqslant x_i \leqslant 1.28,\ \min(f_7) = f_7(0, 0, \cdots, 0) = 0$$

(8) *Generalized Schwefel's Problem* 2.26:

$$f_8(x) = -\sum_{i=1}^{D}(x_i \sin(\sqrt{|x_i|}))$$

$$-500 \leqslant x_i \leqslant 500,\ \min(f_8) = f_8(420.968\,7, 420.968\,7, \cdots, 420.968\,7) = -418.98D$$

(9) *Generalized Rastrigin function*:

$$f_9(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$$

$$-5.12 \leqslant x_i \leqslant 5.12,\ \min(f_9) = f_9(0, \cdots, 0) = 0$$

(10) *Ackley's function*:

$$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$$

$$-32 \leqslant x_i \leqslant 32,\ \min(f_{10}) = f_{10}(0, \cdots, 0) = 0$$

(11) *Generalized Griewank function*:

$$f_{11}(x) = \frac{1}{4\,000}\sum_{i=1}^{D}x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$-600 \leqslant x_i \leqslant 600,\ \min(f_{11}) = f_{11}(0, \cdots, 0) = 0$$

(12) *Penalized Function P8*:

$$f_{12}(x) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\right\} + \sum_{i=1}^{D}u(x_i, 10, 100, 4)$$

$-50 \leqslant x_i \leqslant 50$, $\min(f_{12}) = f_{12}(1, 1, \cdots, 1) = 0$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

(13) *Penalized Function P16*:

$$f_{13}(x) = 0.1\{\sin^2(\pi 3x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$$

$-50 \leqslant x_i \leqslant 50$, $\min(f_{13}) = f_{13}(1, 1, \cdots, 1) = 0$

3.3.2 新系数的设置

受微粒群优化算法的启发,在引力搜索算法的位置更新方程中引入了新的系数 s。而系数 s 对引力搜索算法的优化性能是否有改善,需要通过实验进行验证。s 的取值共有两种类型,一种是取固定值,另一种是变量。固定值分别取 1.2、1、0.8、0.6、0.4、0.2 和 0,共有 7 种情形。变量考虑最基本的形式,其值随迭代次数线性递减,表达式为

$$s = s_{\max} - (s_{\max} - s_{\min})\frac{t}{T} \tag{3.24}$$

其中,t 表示当前的迭代次数;T 表示最大迭代次数;$s_{\max}$ 和 $s_{\min}$ 分别表示 s 的上下界,两者的取值分别为 1 和 0。

方便起见,系数 s 这 8 种取值情况对应的引力搜索算法分别称为 MGSA1、MGSA2、MGSA3、MGSA4、MGSA5、MGSA6、MGSA7 和 MGSA8。根据试验,此时引力搜索算法中另外几个参数设置情况为:$N=50$, $T=1\,000$, $G_0=100$ 和 $\alpha=50$。测试函数取 30 维,即 $D=30$。对每个测试函数,每种算法独立运行 30 次,分别统计最优值、最差值、平均值和标准差等指标,具体结果如表 3.1 所示。

表 3.1　8 种算法的实验结果

函　数	算　法	最优值	最差值	平均值	标准差
f_1	MGSA1	5.276 8e+004	7.603 4e+004	6.439 6e+004	5.989 7e+003
	MGSA2	2.233 1e−036	3.425 3	0.203 8	0.758 9
	MGSA3	2.061 4e−083	6.046 6e−082	2.090 0e−082	1.566 6e−082
	MGSA4	6.502 9e−140	1.137 3e−138	2.614 2e−139	2.049 7e−139
	MGSA5	6.662 7e−195	5.263 1e−192	3.269 4e−193	0
	MGSA6	9.527 6e−210	3.801 7e−205	3.496 7e−206	0
	MGSA7	1.564 5e−208	9.364 8e−204	1.033 9e−204	0
	MGSA8	8.985 8e−211	4.229 9e−206	3.035 1e−207	0
f_2	MGSA1	22.398 6	30.597 5	25.567 4	2.049 7
	MGSA2	5.660 9e−018	0.365 1	0.033 2	0.084 6
	MGSA3	4.719 8e−041	3.610 5e−040	1.350 2e−040	7.202 6e−041
	MGSA4	5.662 7e−069	2.709 1e−068	1.216 7e−068	5.082 9e−069
	MGSA5	1.274 0e−096	5.253 5e−095	1.363 7e−095	1.111 2e−095
	MGSA6	2.926 2e−104	2.822 3e−101	2.545 0e−102	5.398 2e−102
	MGSA7	1.669 2e−103	2.792 7e−101	7.348 1e−102	8.441 3e−102
	MGSA8	2.217 9e−104	6.569 6e−102	5.947 5e−103	1.197 0e−102
f_3	MGSA1	7.860 2e+004	1.698 2e+005	1.174 6e+005	2.206 0e+004
	MGSA2	340.391 0	1.520 6e+003	816.976 9	299.243 6
	MGSA3	2.814 6e−080	1.251 6e−079	6.863 5e−080	2.334 9e−080
	MGSA4	6.771 3e−138	4.046 2e−137	1.876 4e−137	9.468 5e−138
	MGSA5	1.743 0e−192	3.938 7e−189	4.930 4e−190	0
	MGSA6	1.565 1e−203	1.051 5e−199	1.282 1e−200	0
	MGSA7	3.775 3e−203	6.060 1e−198	3.112 9e−199	0
	MGSA8	2.384 8e−205	8.109 8e−200	3.251 6e−201	0

续表

函　数	算　法	最优值	最差值	平均值	标准差
f_4	MGSA1	74.640 5	92.573 4	86.537 7	4.743 1
	MGSA2	3.495 8	7.672 0	5.644 0	1.160 2
	MGSA3	1.047 6e−041	5.301 7e−041	2.316 0e−041	8.946 2e−042
	MGSA4	2.626 5e−070	1.063 5e−069	4.937 8e−070	1.692 1e−070
	MGSA5	7.175 2e−098	7.116 4e−096	2.091 8e−096	1.794 1e−096
	MGSA6	3.922 5e−105	1.160 6e−101	1.567 4e−102	2.548 3e−102
	MGSA7	5.508 8e−103	5.009 3e−101	7.756 4e−102	1.043 9e−101
	MGSA8	1.279 5e−104	1.178 7e−102	3.180 1e−103	3.075 0e−103
f_5	MGSA1	6.168 6e+005	5.794 1e+006	2.466 4e+006	1.252 4e+006
	MGSA2	26.382 5	285.472 5	99.601 7	66.301 9
	MGSA3	28.658 5	28.801 3	28.739 5	0.035 0
	MGSA4	28.706 0	28.823 7	28.779 6	0.031 3
	MGSA5	28.675 0	28.869 6	28.790 9	0.053 2
	MGSA6	28.748 8	28.880 3	28.829 1	0.031 5
	MGSA7	28.670 3	28.901 0	28.844 6	0.053 6
	MGSA8	28.601 8	28.781 5	28.714 9	0.045 2
f_6	MGSA1	51 306	76 368	6.586 4e+004	6.970 9e+003
	MGSA2	0	58	12.566 7	14.277 3
	MGSA3	0	0	0	0
	MGSA4	0	0	0	0
	MGSA5	0	0	0	0
	MGSA6	0	0	0	0
	MGSA7	0	0	0	0
	MGSA8	0	0	0	0

续表

函数	算法	最优值	最差值	平均值	标准差
f_7	MGSA1	58.123 9	110.614 5	84.231 2	14.115 4
	MGSA2	0.012 8	0.290 4	0.102 5	0.071 4
	MGSA3	8.469 3e-007	6.238 7e-005	2.347 7e-005	1.765 1e-005
	MGSA4	2.513 9e-006	1.107 1e-004	2.304 7e-005	2.498 4e-005
	MGSA5	1.246 0e-008	6.677 8e-005	1.707 5e-005	1.977 0e-005
	MGSA6	1.023 0e-007	8.196 3e-005	2.362 8e-005	2.244 6e-005
	MGSA7	2.342 0e-007	1.154 8e-004	2.744 9e-005	3.114 8e-005
	MGSA8	1.958 7e-007	8.517 2e-005	2.162 9e-005	1.980 5e-005
f_8	MGSA1	-4.179 9e+003	-2.417 3e+003	-3.057 3e+003	379.264 8
	MGSA2	-3.761 1e+003	-2.113 9e+003	-2.723 7e+003	413.777 3
	MGSA3	-3.632 6e+003	-1.703 9e+003	-2.528 5e+003	481.237 1
	MGSA4	-3.806 7e+003	-1.678 9e+003	-2.353 7e+003	523.072 0
	MGSA5	-3.144 6e+003	-1.686 8e+003	-2.388 7e+003	379.709 1
	MGSA6	-4.989 6e+003	-1.615 2e+003	-2.550 3e+003	740.189 4
	MGSA7	-4.085 6e+003	-1.759 7e+003	-2.388 2e+003	523.517 9
	MGSA8	-3.494 1e+003	-1.967 8e+003	-2.673 3e+003	332.447 1
f_9	MGSA1	268.104 8	342.344 6	295.505 4	15.012 5
	MGSA2	9.949 6	31.838 6	19.003 7	4.475 2
	MGSA3	0	0	0	0
	MGSA4	0	0	0	0
	MGSA5	0	0	0	0
	MGSA6	0	0	0	0
	MGSA7	0	0	0	0
	MGSA8	0	0	0	0

续表

函　数	算　法	最优值	最差值	平均值	标准差
f_{10}	MGSA1	12.150 3	15.599 3	13.804 0	0.837 8
	MGSA2	4.440 9e−015	1.154 6e−014	7.164 6e−015	1.790 6e−015
	MGSA3	8.881 8e−016	8.881 8e−016	8.881 8e−016	0
	MGSA4	8.881 8e−016	8.881 8e−016	8.881 8e−016	0
	MGSA5	8.881 8e−016	8.881 8e−016	8.881 8e−016	0
	MGSA6	8.881 8e−016	8.881 8e−016	8.881 8e−016	0
	MGSA7	8.881 8e−016	8.881 8e−016	8.881 8e−016	0
	MGSA8	8.881 8e−016	8.881 8e−016	8.881 8e−016	0
f_{11}	MGSA1	484.718 7	700.292 5	591.027 9	59.020 8
	MGSA2	30.767 6	62.868 1	44.502 2	8.458 4
	MGSA3	0	0	0	0
	MGSA4	0	0	0	0
	MGSA5	0	0	0	0
	MGSA6	0	0	0	0
	MGSA7	0	0	0	0
	MGSA8	0	0	0	0
f_{12}	MGSA1	5.075 5e+004	2.318 4e+007	4.654 0e+006	5.081 7e+006
	MGSA2	0.022 4	3.152 6	1.197 2	0.841 8
	MGSA3	0.177 1	0.355 3	0.274 5	0.043 2
	MGSA4	0.230 6	0.456 7	0.327 1	0.049 3
	MGSA5	0.321 7	0.657 8	0.469 4	0.060 4
	MGSA6	0.368 8	0.722 7	0.555 5	0.087 0
	MGSA7	0.481 2	0.812 9	0.639 3	0.075 7
	MGSA8	0.069 5	0.648 9	0.278 7	0.152 1

续表

函　数	算　法	最优值	最差值	平均值	标准差
f_{13}	MGSA1	6.335 1e+006	4.668 5e+007	2.388 2e+007	1.196 8e+007
	MGSA2	0.265 1	27.000 5	12.456 8	7.868 2
	MGSA3	1.681 0	2.289 7	1.978 2	0.146 2
	MGSA4	1.804 9	2.420 3	2.064 6	0.156 2
	MGSA5	2.088 0	2.876 1	2.482 9	0.170 4
	MGSA6	2.231 4	2.975 5	2.795 8	0.185 2
	MGSA7	2.484 8	2.986 1	2.888 7	0.118 1
	MGSA8	0.649 8	1.544 3	1.105 2	0.211 8

根据表 3.1 的实验结果可以明显看出，系数 s 的引入对引力搜索算法的优化性能有着显著的影响。从最优值、最差值、平均值和标准差这 4 个方面对这 8 种算法进行比较，可以得到以下结论。对于函数 f_1、f_2、f_3、f_4、f_5、f_7、f_{12}和 f_{13}而言，MGSA8 的求解结果明显优于其他几种算法，优化性能显著；对于函数 f_6、f_9、f_{10}和 f_{11}而言，MGSA8、MGSA3、MGSA4、MGSA5、MGSA6 和 MGSA7 这几种算法都获得了最好的优化结果，其结果远优于 MGSA1 和 MGSA2 的结果。对于函数 f_8 而言，这几种算法的求解结果均不够理想，相对来说 MGSA1 的结果最好，稍优于 MGSA8 的结果。因此，就总体情况而言，MGSA8 的求解结果最佳，优化性能最好。

在 MGSA8 中，设置系数 s 的策略是使其取值随迭代次数线性递减，而在其他几种算法中，系数 s 的取值都是固定值。通过上述实验发现，系数 s 采用动态调整的策略效果更好些。在 MGSA8 中，开始时 s 的取值较大，然后逐步减小，最后趋向于 0。在算法运行的初期，s 的值较大有利于全局探索，而在算法运行的后期，s 的值较小有利于局部开发。全局探索和局部开发的能力对智能优化算法的优化性能有着极其重要的影响。全局探索主要目的是对解空间进行更全面的搜索，希望发现更多的未知区域，而局部开发主要目的是对已知区域进行更为精细的搜索，希望获得质量更好的新解。尽管这两者的最终目的都是希望找到全局最优解，但具体实现方法不同。而计算时间是有限的，因此会在如何利用这有限的计算时间上

产生矛盾(计算时间在算法中具体可以通过迭代次数等指标体现)。例如,算法过多地进行全局搜索而削弱其局部开发能力,算法在运行后期会出现局部搜索能力不强和得到的解质量不高等问题。从智能优化算法的发展历史来看,任何一种新方法,一方面具有良好的优化潜力,另一方面探索和开发能力往往很难达到有效的平衡,有待进一步完善。引力搜索算法作为一种新型智能优化算法也同样如此。从现有研究文献来看,算法的全局探索能力较强、而局部开发能力较弱,无法实现全局优化。一个好的智能优化算法的关键在于全局探索和局部开发能力的平衡。例如,要求算法在执行的早期具有较好的全局探索能力,能够对解空间进行比较全面的搜索,而在算法执行的后期具有较好的局部开发能力,能够对已有区域进行比较精细的搜索。实现算法全局探索和局部开发能力平衡的方法有很多种,需要根据算法自身的特点选择可行且有效的方法。就引力搜索算法而言,通过和微粒群优化算法的比较,在位置更新方程中引入了新的系数 s。同时借助实验,确定了系数 s 的取值方法——随迭代次数线性递减。系数 s 这种动态变化策略,实现了引力搜索算法的全局探索和局部开发能力的平衡,有效解决了基本引力搜索算法全局探索能力较强而局部开发能力较弱的问题,显著提高了基本引力搜索算法的优化性能。在本章中,如无特别说明,采用 MGSA 表示 MGSA8。

3.3.3 比较实验

为进一步验证改进引力搜索算法 MGSA 的优化性能,采用 3.4.1 节给出的 13 个标准测试函数进行实验,并将其与基本引力搜索算法 GSA 和微粒群优化算法 SPSO2011[99]的求解结果进行比较。目前,微粒群优化算法是求解单目标连续优化问题的一种典型的智能优化算法,而 SPSO2011 是最新的一种微粒群优化算法。因此,选择 SPSO2011 作为比较算法具有一定的代表性。

这 3 种算法的群体规模和最大迭代次数设置相同,两者取值分别为 50 和 1 000。改进引力搜索算法 MGSA 的其他参数设置和 3.4.2 节中的设置一样,基本引力搜索算法 GSA 中的其他参数设置采用文献[63]中的推荐值,微粒群优化算法 SPSO2011 的其他参数设置采用文献[99]中的参考值。每个测试函数的维数为 30,每种算法独立运行 30 次,分别记录其最优值、最差值、平均值和标准差的计算结果,实验结果如表 3.2 所示,其中 MGSA 的结果来自表 3.1 中的实验结果。

表 3.2　3 种算法的实验结果

函　数	算　法	最优值	最差值	平均值	标准差
f_1	SPSO2011	4.737 3e−027	1.804 8e−024	4.481 1e−025	5.307 3e−025
	GSA	1.258 1e−017	3.568 0e−017	2.204 5e−017	7.102 3e−018
	MGSA	8.985 8e−211	4.229 9e−206	3.035 1e−207	0
f_2	SPSO2011	0.127 0	2.299 4	0.876 1	0.588 1
	GSA	1.687 0e−008	2.833 0e−008	2.267 7e−008	3.306 2e−009
	MGSA	2.217 9e−104	6.569 6e−102	5.947 5e−103	1.197 0e−102
f_3	SPSO2011	0.182 5	1.541 6	0.575 0	0.309 3
	GSA	91.875 6	424.573 3	217.405 2	81.390 2
	MGSA	2.384 8e−205	8.109 8e−200	3.251 6e−201	0
f_4	SPSO2011	0.059 5	5.772 9	1.570 7	1.194 1
	GSA	1.768 9e−009	0.268 7	0.009 0	0.049 1
	MGSA	1.279 5e−104	1.178 7e−102	3.180 1e−103	3.075 0e−103
f_5	SPSO2011	19.803 9	138.276 3	34.649 6	28.215 6
	GSA	25.813 5	26.336 8	26.085 1	0.151 0
	MGSA	28.601 8	28.781 5	28.714 9	0.045 2
f_6	SPSO2011	0	2	0.400 0	0.621 5
	GSA	0	0	0	0
	MGSA	0	0	0	0
f_7	SPSO2011	0.002 2	0.008 2	0.005 3	0.001 6
	GSA	0.008 9	0.212 4	0.032 2	0.036 5
	MGSA	1.958 7e−007	8.517 2e−005	2.162 9e−005	1.980 5e−005
f_8	SPSO2011	−8.857 8e+003	−5.051 1e+003	−6.780 4e+003	910.610 4
	GSA	−4.104 8e+003	−2.041 0e+003	−2.765 7e+003	473.961 2
	MGSA	−3.494 1e+003	−1.967 8e+003	−2.673 3e+003	332.447 1

续表

函 数	算 法	最优值	最差值	平均值	标准差
f_9	SPSO2011	11.939 5	82.901 1	28.196 4	17.060 0
	GSA	10.944 5	28.853 8	18.970 5	4.464 4
	MGSA	0	0	0	0
f_{10}	SPSO2011	2.211 6e－013	2.316 8	0.794 3	0.754 1
	GSA	2.688 8e－009	4.504 9e－009	3.566 5e－009	4.224 6e－010
	MGSA	8.881 8e－016	8.881 8e－016	8.881 8e－016	0
f_{11}	SPSO2011	2.220 4e－015	0.031 9	0.008 6	0.009 3
	GSA	1.325 1	7.198 9	3.979 4	1.510 3
	MGSA	0	0	0	0
f_{12}	SPSO2011	2.808 6e－018	0.231 2	0.035 4	0.070 7
	GSA	7.312 8e－020	0.103 7	0.031 1	0.048 3
	MGSA	0.069 5	0.648 9	0.278 7	0.152 1
f_{13}	SPSO2011	9.489 9e－020	0.054 8	0.007 3	0.013 0
	GSA	1.297 7e－018	0.011 0	3.662 5e－004	0.002 0
	MGSA	0.649 8	1.544 3	1.105 2	0.211 8

根据表 3.2，从最优值、最差值、平均值和标准差这几个方面对 SPSO2011、GSA 和 MGSA 这 3 种算法的求解结果进行比较。在求解函数 f_1、f_2、f_3、f_4、f_7、f_9、f_{10}和 f_{11}时，MGSA 表现最好，优化效果最佳，性能明显优于其他两种算法；在求解函数 f_6 时，MGSA 和 GSA 都获得了最优结果，这两种算法均好于 SPSO2011；在求解函数 f_5 时，3 种算法中 SPSO2011 表现最好，MGSA 次之，GSA 最差；在求解函数 f_8 时，这 3 种算法表现得都不太理想，相对而言，SPSO2011 的计算结果最好；在求解函数 f_{12} 和 f_{13} 时，SPSO2011 和 GSA 的得到的结果稍好于 MGSA 的结果。在求解这 13 个典型的测试函数时，MGSA 在其中 8 个函数上获得的结果最好，在其中 1 个函数上和 GSA 的结果一样好，并优于 SPSO2011 的结果，在其余 4 个函数上，稍劣于 SPSO2011 或 GSA 的结果。因此，对大多数测试函数而

言，这 3 种算法中 MGSA 具有最好的优化性能，是一种可行有效的智能优化算法。

MGSA 的良好的优化性能主要得益于在位置更新方程中系数 s 的引入，并且取值采用随迭代次数线性递减的动态调整策略，实现了算法全局探索和局部开发能力的平衡。从现有的研究结果可以看出，基本引力搜索算法具有较好的全局探索能力，但是局部开发能力较弱，求解精度不高，易陷入局部极值。最基本的解决办法是增加算法的迭代次数，但在实际计算过程中时间是有限的，需要采取其他解决方法。通过系数 s 的引入，改进的引力搜索算法有效地解决了基本引力搜索算法全局探索能力强而局部开发能力弱的问题，显著提高了基本引力搜索算法的优化性能。

虽然引力搜索算法和微粒群优化算法都是采用速度和位置的计算模型，但是这两种算法存在本质的区别：

(1) 引力搜索算法源于对万有引力定律的模拟，而微粒群优化算法是对鸟群觅食行为的模拟。

(2) 在引力搜索算法中考虑个体质量，并且个体质量是算法的一个重要组成部分，而在微粒群优化算法中不考虑个体质量。

(3) 在引力搜索算法中，整个群体是在引力的作用下逐步向最优位置逼近，而在微粒群优化算法整个群体是在个体和群体搜索经验作用下逐渐向最优位置靠近。

为进一步分析 SPSO2011、GSA 和 MGSA 的优化性能，这 3 种算法在所有测试函数上的寻优曲线在图 3.1 至图 3.13 中给出。

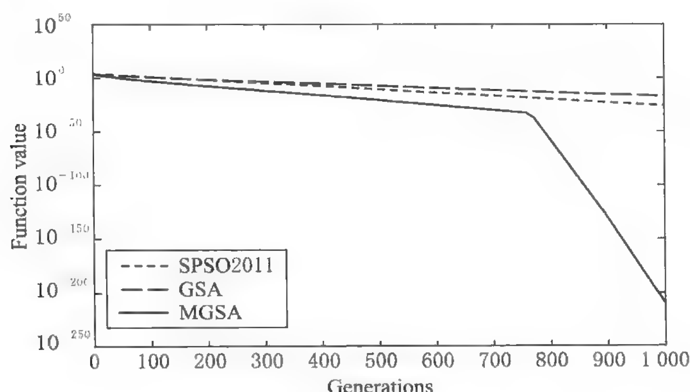


图 3.1　f_1 优化曲线对比图

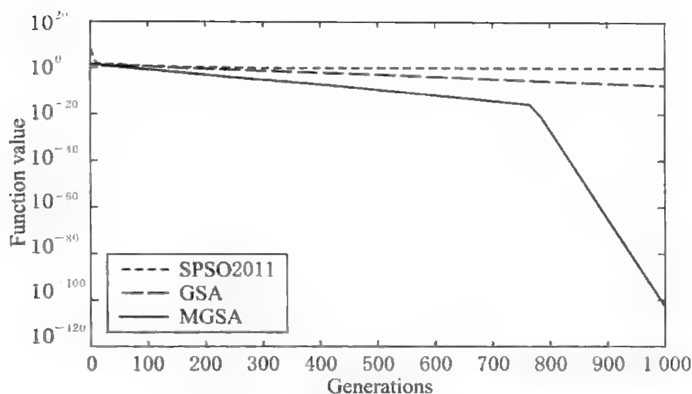


图 3.2　f_2 优化曲线对比图

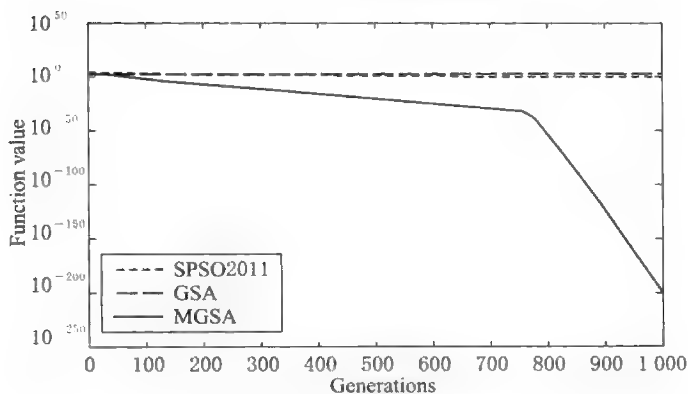


图 3.3　f_3 优化曲线对比图

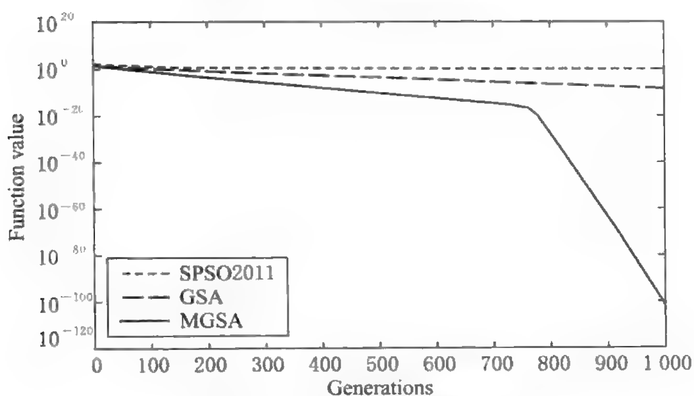


图 3.4　f_4 优化曲线对比图

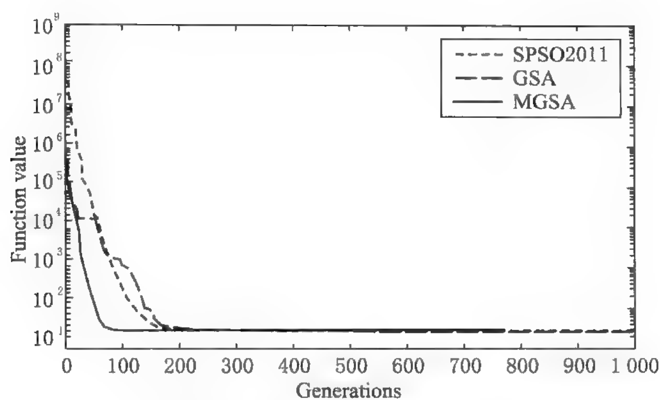


图 3.5　f_5 优化曲线对比图

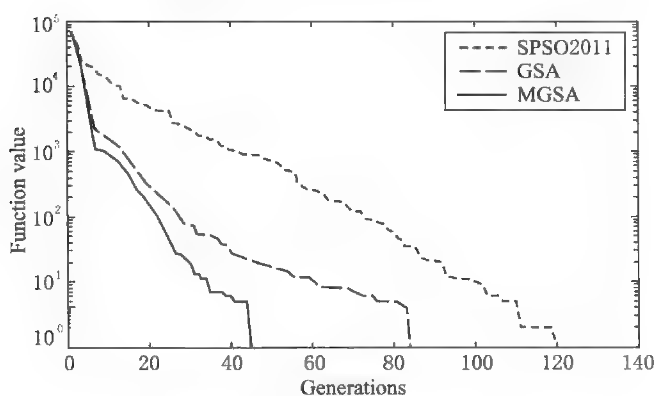


图 3.6　f_6 优化曲线对比图

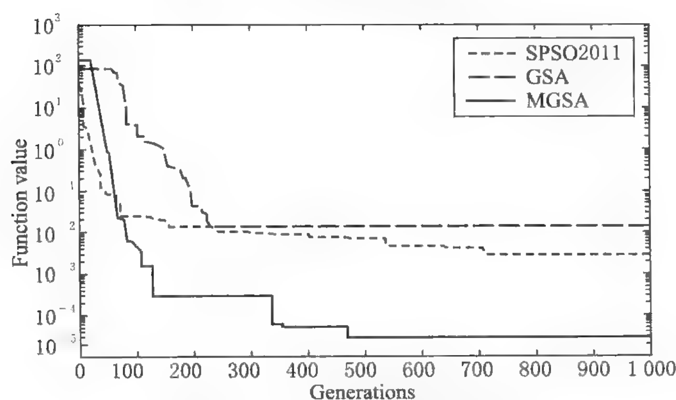


图 3.7　f_7 优化曲线对比图

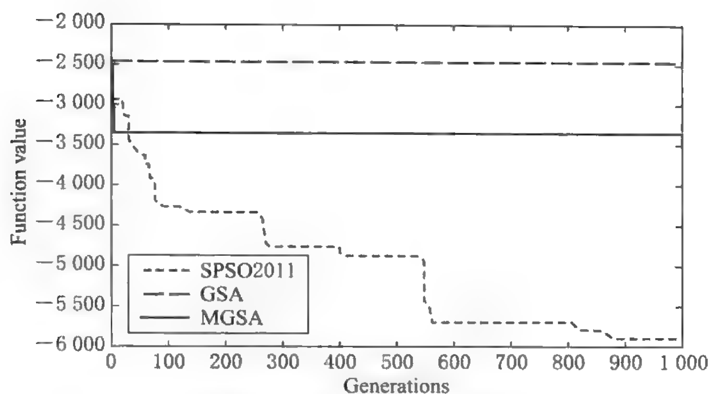


图 3.8 f_8 优化曲线对比图

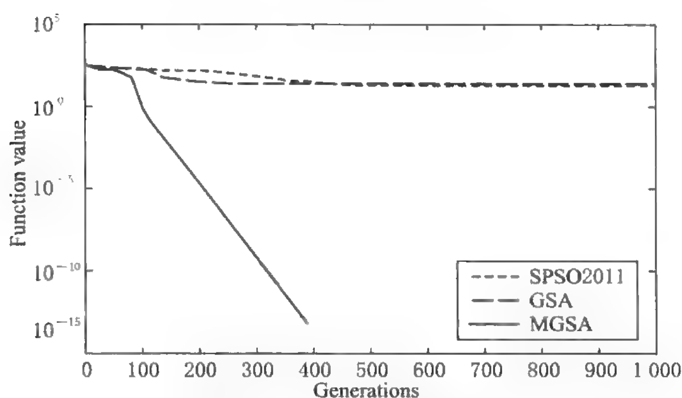


图 3.9 f_9 优化曲线对比图

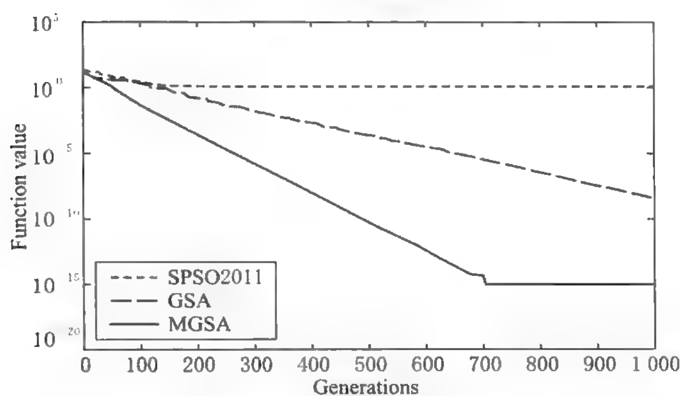


图 3.10 f_{10} 优化曲线对比图

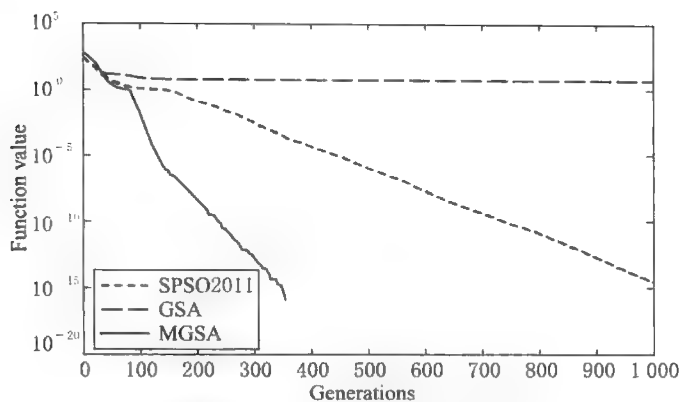


图 3.11　f_{11} 优化曲线对比图

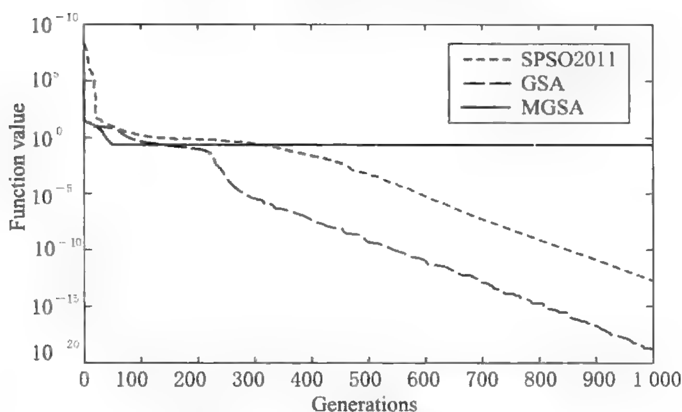


图 3.12　f_{12} 优化曲线对比图

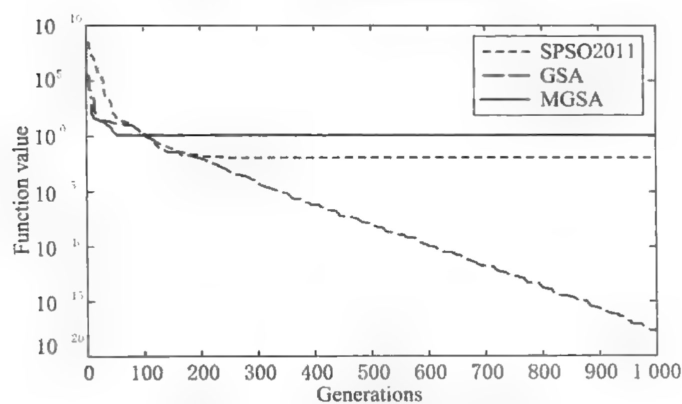


图 3.13　f_{13} 优化曲线对比图

表 3.2 中的计算结果已经说明，对大多数测试函数而言，在 3 种算法中改进引力搜索算法的求解精度最高。同时从这些函数优化曲线比较示意图同样可以看出，改进引力搜索算法在大多数情况下具有最好的优化精度。此外，从这些函数的优化曲线图中也可以看出，除少数函数外，改进引力搜索算法在这 3 种算法中具有最快的寻优速度。

为进一步分析这 3 种算法的优化速度，采用文献[100]中给出的进展度量指标 P 进行评价。进展度量指标 P 可以用于分析单目标优化中智能优化算法的收敛速率，是一种相对而非绝对的衡量标准。其具体计算方法如下：

$$P = \ln\sqrt{\frac{f_{\max}(0)}{f_{\max}(t)}} \tag{3.25}$$

其中，$f_{\max}(0)$表示初始群体中的最优适应度函数值，$f_{\max}(t)$表示第 t 次迭代时群体中的最优适应度函数值。图 3.14 至图 3.26 给出了这 3 种算法进展度量指标随迭代次数变化的示意图。

从这 3 种算法进展度量 P 的比较示意图中可以发现，对大多数测试函数而言，改进引力搜索算法的寻优速度最快。因此，通过上述一系列的实验比较，可以得到以下结论：改进引力搜索算法不仅求解精度高，而且优化速度快。

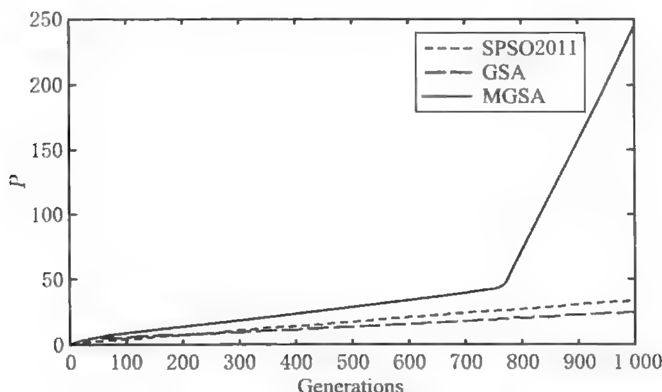


图 3.14　f_1 优化过程中的 P 值

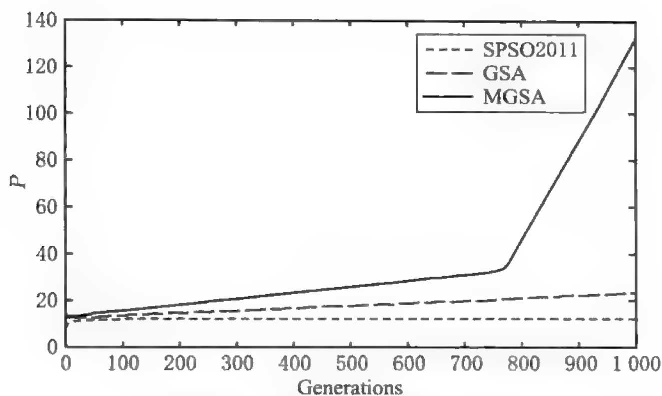


图 3.15　f_2 优化过程中的 P 值

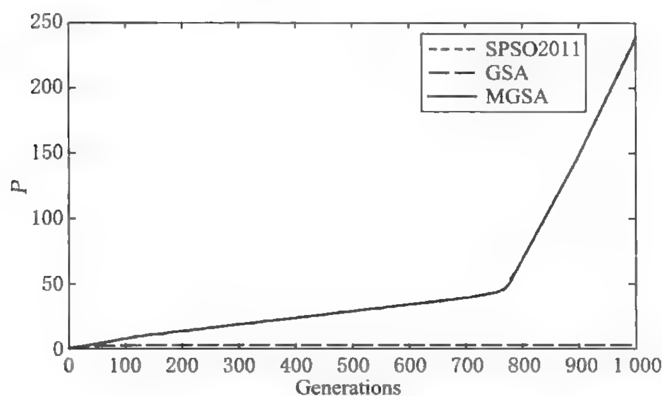


图 3.16　f_3 优化过程中的 P 值

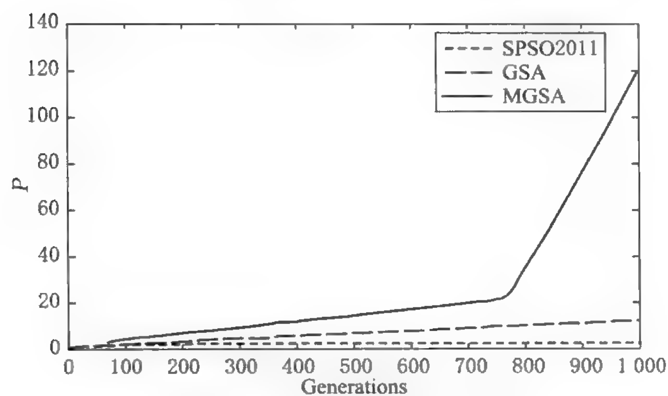


图 3.17　f_4 优化过程中的 P 值

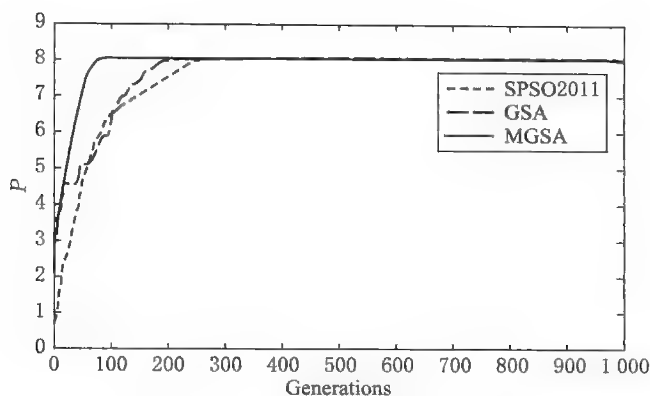


图 3.18 f_5 优化过程中的 P 值

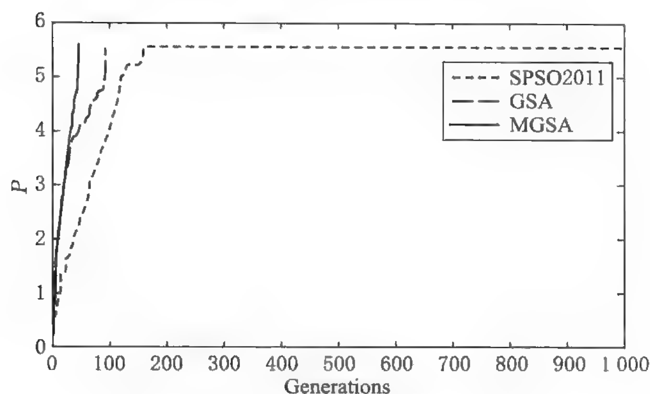


图 3.19 f_6 优化过程中的 P 值

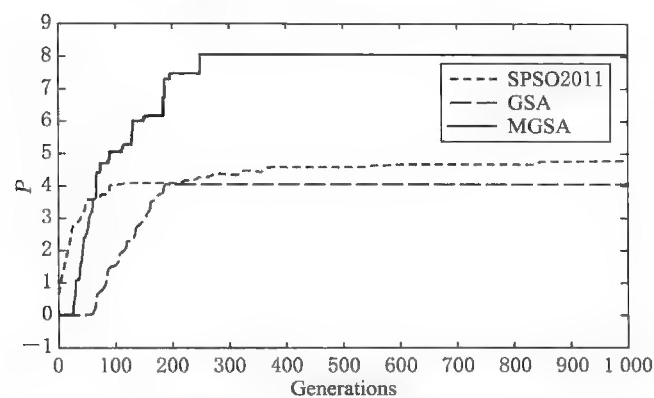


图 3.20 f_7 优化过程中的 P 值

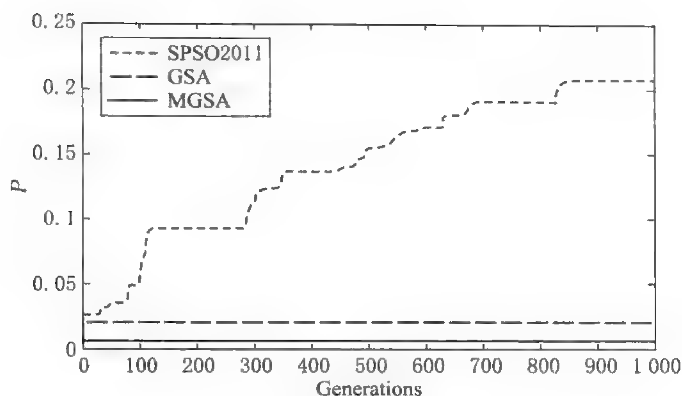


图 3.21　f_8 优化过程中的 P 值

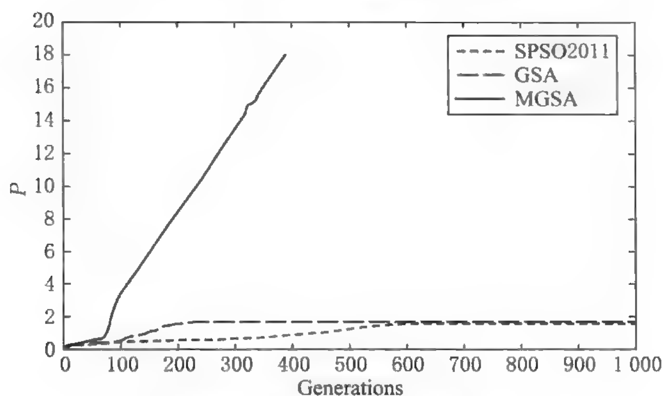


图 3.22　f_9 优化过程中的 P 值

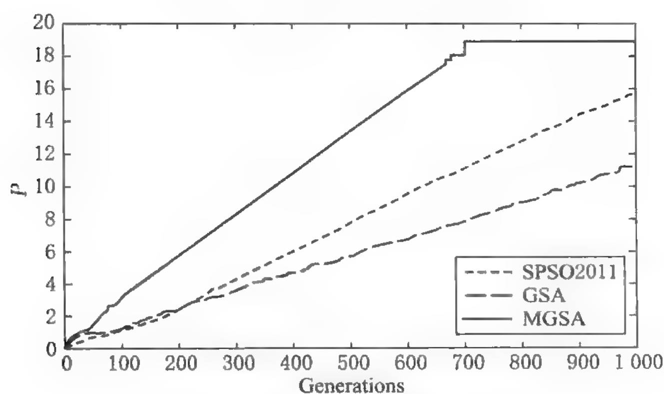


图 3.23　f_{10} 优化过程中的 P 值

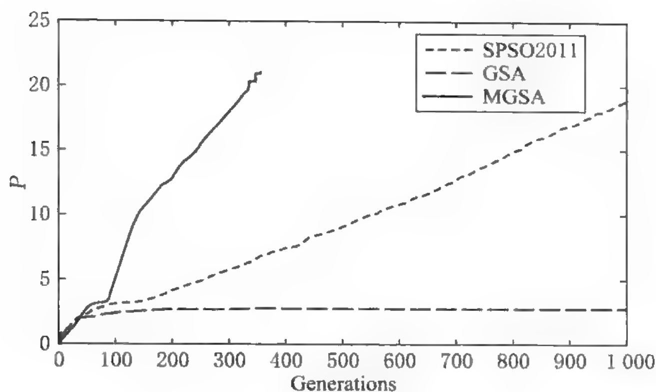


图 3.24　f_{11} 优化过程中的 P 值

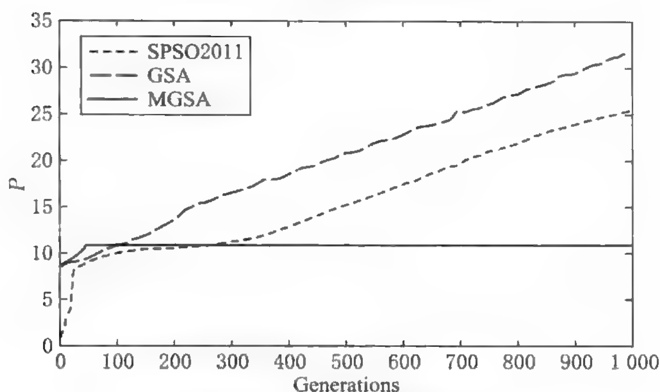


图 3.25　f_{12} 优化过程中的 P 值

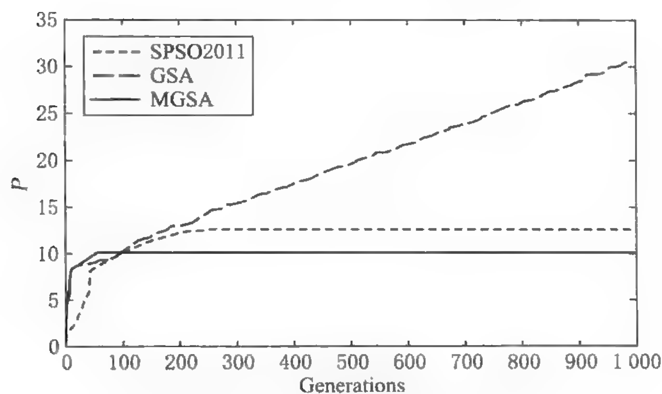


图 3.26　f_{13} 优化过程中的 P 值

第四章　多目标连续优化的引力搜索算法

最早出现的多目标优化问题，可以追溯到1772年，当时Franklin提出了多目标矛盾如何进行协调的问题。但是一般认为多目标优化问题最早是由法国经济学家和社会学家Pareto于1896年提出，Pareto从政治经济学角度出发，把许多不好比较的目标归纳为多目标优化问题。自此，多目标优化问题受到越来越多的研究人员的关注[101]。

目前多目标优化问题广泛应用于生产管理、经济决策、工程设计和国防建设等领域，如物流运输、投资预算、城市布局、工业制造和导弹设计等。如此看来，多目标优化问题在现实生活中大量存在着。实际上，如果单目标优化问题在解决实际问题时很重要的话，那么多目标优化应该更加重要，因为在处理一个问题时，往往要考虑的目标不止一个。

通常情况下，多目标优化问题的各子目标之间是相互矛盾的，如一个子目标性能的提高往往会引起另一个或几个子目标性能的降低。多目标优化问题的最优解一般情况下并不唯一，而是由一组Pareto最优解构成的集合。因此，设计有效的多目标优化问题的解法具有很大的挑战性。

目前，求解多目标优化问题的方法主要分为传统优化算法和智能优化算法。传统方法是将多目标优化问题转换为单目标优化问题进行求解，典型的算法有加权法、ε约束法和目标规划法等。这些经典算法中，一些成熟的单目标优化算法可直接被使用，使得传统算法具有一定的吸引力和优越性，但也存在一些不足。如加权法无法有效确定权重系数，对Pareto最优前端比较敏感，不能较好地处理前端的凹部，从而限制了其进一步的应用；ε约束法设置合适的ε值往往需要搜索空间的先验知识，但是这些先验知识绝大多数情况下是未知的。和ε约束法一样，目标规划法要求决策者事先设置目标值，但由于缺乏足够的先验知识，对这个目标值的设立具有一定的主观性。此外，传统算法共同的一个不足之处就是要获得多个

Pareto最优解要多次运行优化方法，而这些求解过程往往相互独立，它们之间的信息无法实现共享，导致计算资源的浪费。

20世纪80年代中期，智能优化算法开始用于求解多目标优化问题，并受到广泛的关注。实际上，早在1967年Rosenberg就在其博士论文中就提出可以采用遗传算法求解多目标优化问题，但他没有给出具体的实现方法。Schaffer在1985年提出了矢量评价遗传算法，开创了智能优化算法求解多目标优化问题的先河。智能优化算法在处理多目标优化问题时具有一定的优势：第一，通常情况下，智能优化算法都是采用基于群体的搜索方式，运行算法一次可以获得多个Pareto最优解；第二，对目标的最优均衡面的形状和连续性不敏感，可以较好地逼近不连续或非凸性的均衡面；第三，算法中都存在优化信息传递机制，个体可以利用共享的优化信息指导其搜索行为[102,103]。

经过近30年的发展，已经出现了很多求解多目标优化问题的智能优化算法。有以非支配排序和小生境技术为特征的算法，包括MOGA、NSGA和NPGA等；有基于精英保留策略的多目标优化问题的处理方法，包括SPEA、SPEA2、PAES、PESA、PESA-Ⅱ和NSGA-Ⅱ等。

通常情况下，利用智能优化算法求解多目标优化问题时希望达到下面两点：

(1) 算法得到的Pareto最优前端和优化问题真实的Pareto最优前端要尽可能的接近；

(2) 算法得到的Pareto最优前端的分布性好，尽可能地呈现均匀分布。

同时满足上述两个目标，对任何一个求解多目标优化问题的算法都具有挑战性。迄今为止，虽然出现了很多用于多目标优化的算法，为求解该类问题带来了希望，但是问题远未解决。因此，设计新的求解多目标优化问题的算法仍然是非常重要的研究课题[101]。

2010年，Hassanzadeh等提出了多目标引力搜索算法，第一次实现了引力搜索算法和多目标优化问题的结合[75]。2011年，Nobahari等给出了一种非劣分类的引力搜索算法[76]；Gonzaalez-alvarez等给出一种求解图像识别问题的多目标引力搜索算法[104]；Rubio-largo等提出一种优化静态路由选择和波长分配问题的多目标引力搜索算法[105]。这些算法为多目标优化问题的求解提供了新的思路和方法，但从实验结果来看，算法性能有待进一步改进。此外，这些算法缺少理论分析。

本章采用非支配排序、拥挤距离计算和精英保留策略等方法，提出一种新的

多目标引力搜索算法(Novel Multiobjective Gravitational Search Algorithm，NMGSA)。从理论角度分析算法的收敛性，从实验角度验证算法的优化性能。

4.1 算法设计

在利用引力搜索算法求解多目标优化问题时，必须考虑以下三个关键问题：

(1) 如何避免算法早熟收敛，保证算法产生的群体向 Pareto 最优前端搜索；

(2) 如何避免个体在局部堆积，获得具有良好分布的 Pareto 最优解集；

(3) 如何结合收敛性和分布情况，定义个体的质量函数。

4.1.1 基本概念

在多目标优化问题中，对不同的子目标可能会有不同的要求。有的要求最大化子目标函数，有的要求最小化子目标函数。归纳起来，共有 3 种情况：(1)最小化所有子目标函数；(2)最大化所有子目标函数；(3)部分子目标函数最小化，其他子目标函数最大化。为处理方便，可将各个子目标的优化问题都转换为最小化或最大化问题。例如将最大化问题转换为最小化问题，该过程可简单地用下式表示：

$$\max f_i(x) = -\min(-f_i(x)) \tag{4.1}$$

其中，$i = 1, 2, \cdots, m$，m 表示子目标函数的个数。

本章若无特别说明，均考虑最小化的多目标优化问题。

定义 4.1 一个具有 m 个目标函数和 n 个决策变量的多目标优化问题可以表示为

$$\begin{aligned} &\min f(x) = (f_1(x), f_2(x), \cdots, f_m(x)) \\ &s.t. \quad x \in X \subset R^n \end{aligned} \tag{4.2}$$

多目标优化问题的最优解一般被称为 Pareto 最优解，是由 Pareto 于 1896 年提出，并被命名为 Pareto 最优解。

定义 4.2 给定一个多目标优化问题 $\min f(x)$，若 $x^* \in X$，且不存在其他的 $\bar{x}^* \in X$ 使得 $f_j(x^*) \geqslant f_j(\bar{x}^*)(j = 1, 2, \cdots, m)$ 成立，且其中至少一个为严格不等式，则称 x^* 是多目标优化问题 Pareto 最优解。

Pareto 最优解有时也称为非劣解或有效解。由上述定义可以看出，满足 Pareto 最优解条件的解可能不是一个，而是多个。

定义 4.3 所有 Pareto 最优解组成的集合称为 Pareto 最优解集，并用 P_{true} 表示。

为更好地理解 Pareto 最优解集，可以讨论其在目标函数空间中的表现形式。

定义 4.4 Pareto 最优解集中的解对应的目标函数值组成的集合称为 Pareto 最优前端或 Pareto 最优边界，并用 PF_{true} 表示。

引力搜索算法是基于群体智能的方法，分析个体之间的关系比较重要。

定义 4.5 假设 p 和 q 是群体中任意两个不同的个体，若称 p 支配 q，则必须满足以下两个条件：

(1) $\forall k \in \{1, 2, \cdots, m\}$, $f_k(p) \leqslant f_k(q)$，即对所有的子目标，p 不比 q 差；

(2) $\exists l \in \{1, 2, \cdots, m\}$, $f_k(p) < f_k(q)$，即至少存在一个子目标，使 p 比 q 好。

这里，p 为支配的，q 为被支配的，可以表示成 $p > q$。

在用引力搜索算法求解多目标优化问题时，利用辅助群体来存储逐代积累产生的非劣个体，用 $P_{known}(t)$ 表示，t 表示迭代次数。当 $t=0$ 时，$P_{known}(0)$ 定义成空集，用 P_{known} 表示算法停止时最终的 Pareto 最优解集。在优化过程中，对应于算法当前得到的群体，用 $P(t)$ 表示。在第 t 次迭代时，将当前群体 $P(t)$ 加入到已有的非劣解集 $P_{known}(t-1)$ 中，其中的每个个体都要经过过滤检测，劣的个体被去除，剩下的个体构成 $P_{known}(t)$。重复上述过程直到算法结束，如此形成算法的 Pareto 最优解集，并分别用 $PF_{known}(t)$ 和 PF_{known} 表示 $P_{known}(t)$ 和 P_{known} 的 Pareto 最优前端。

4.1.2 非支配排序方法

在处理多目标优化问题时，要对群体中个体间的非劣关系进行分析。本章采用非支配排序的方法，将整个群体分成不同的层级[103]。具体方法如下：当前所有个体中没有被其他任何个体所支配的个体为 Pareto-占优（非支配）的，定义该个体的 rank = 1，所有 Pareto-占优个体的集合是第一层 Pareto-占优集合；然后将这些个体从当前的群体中去除，对余下的群体按照上述的方法产生第二层 Pareto-占优集合。以此类推，将群体中所有的个体都进行排序[103, 106]。图 4.1 给出了一种分层示意图，假设群体规模为 7，目标函数个数为 2。

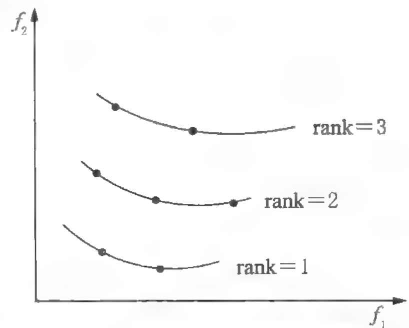


图 4.1 群体非劣分层示意图

当所有个体被分成不同层级后，算法将优先利用层级高（rank 值小）的个体产生新的个体，引导群体向 Pareto 最优解集区域靠近。在保证群体向非劣解集方向不断逼近的同时，还要求整个群体尽可能的分布均匀。

4.1.3 拥挤距离计算

为描述群体的分布情况，可以利用拥挤距离刻画个体间的聚集程度。一般情况下，拥挤距离大的个体其聚集密度小。一个个体的拥挤距离，可以通过在目标空间中同一层上与其相邻的两个个体在每个子目标上的距离之和来计算。如图 4.2 所示，给定一个多目标优化问题，共有两个子目标，分别用 f_1 和 f_2 表示。第 i 个个体的拥挤距离是与其相邻的第 $i-1$ 个个体和第 $i+1$ 个个体在两个子目标 f_1 和 f_2 上的距离之和，即图中实线矩形长与宽的和。在实际计算时，考虑到各个子目标函数值的取值范围的差异，可作归一化处理。设 $I[i]_{distance}$ 表示第 i 个个体的拥挤距离，$I[i].k$ 表示第 i 个个体在子目标 k 上的函数值。通常情况下，当有 m 个子目标函数时，个体 i 的拥挤距离[103, 106]为

$$I[i]_{distance} = \sum_{k=1}^{m} \frac{I[i+1].k - I[i-1].k}{f_k^{\max} - f_k^{\min}} \tag{4.3}$$

其中，$f_k^{\max}$ 和 $f_k^{\min}$ 分别表示第 k 子目标函数值的最大值和最小值。

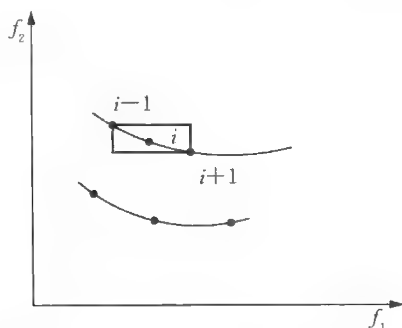


图 4.2 同一层个体之间的拥挤距离的计算

通过非支配排序和拥挤距离的计算，群体中所有个体都有两个特征量：排序号和拥挤距离。两个个体进行比较时，如果非劣层级不同，那么取层级高（rank 数

值小)的个体。否则,如果两个个体在同一层级,那么取拥挤距离大(聚集密度小)的个体。在此基础上,设 $I[i]_{rank}$ 和 $I[i]_{distance}$ 分别表示第 i 个个体进行非支配排序和拥挤距离计算的结果,可以得到两个个体 i 和 j 间的偏序关系 $>_n$:

$$i >_n j = \begin{cases} I[i]_{rank} < I[j]_{rank} \\ I[i]_{rank} = I[j]_{rank} \quad and \quad I[i]_{distance} > I[j]_{distance} \end{cases} \tag{4.4}$$

4.1.4 精英保留策略

为能够引导群体向 Pareto 最优解集区域靠近并保证算法解集分布的均匀性,采用精英保留策略[103,106]。将第 t 代得到的规模为 N 的新群体和算法现有的 Pareto 最优解集 $P_{known}(t-1)$ 进行合并,得到规模为 $2N$ 的群体。根据偏序关系,在合并后的群体中逐一选择优胜个体直到数量达到 N,从而产生第 t 代的 Pareto 最优解集 $P_{known}(t)$。在 $P_{known}(t)$基础上形成新一轮的优化操作,包括计算引力,更新加速度、速度和位置。具体操作过程如图 4.3 所示。

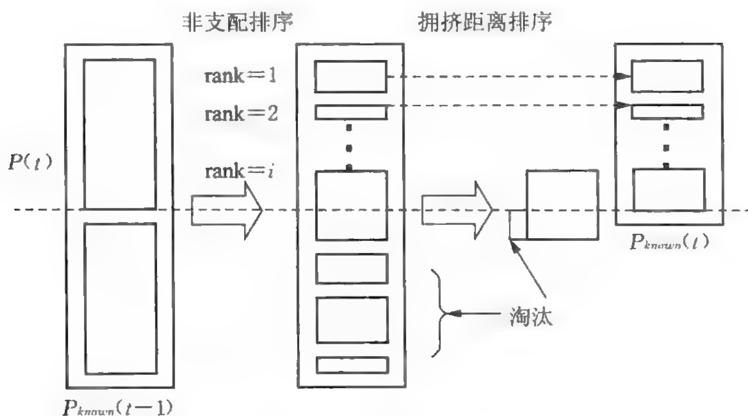


图 4.3 在合并后的群体中选择 N 个个体

这种策略的优点是将连续两代的群体合并,使得群体选择的范围扩大,保证群体具有较好的多样性。根据偏序关系,将优良的个体保存下来。但研究发现,如果算法连续采用这种策略,尽管能保证群体分布的均匀性,但会产生退化现象,即新产生的个体劣于刚被淘汰的个体[102]。文献[107]提出一种局部搜索方

法来改进算法寻优性能，利用线性加权法将多目标优化问题转换为单目标优化问题进行求解，但是权重系数的确定方法有待进一步完善。文献[108]提出一种 ε-近似 Pareto 集的概念来处理退化问题，但选择合适的 ε 取值较为困难。本章提出一种新的处理方法，具体过程如下所述。建立一个被淘汰的个体集合 DA(Deleted Archive, DA)，DA 规模最大为 N，初始值就设为算法的初始群体。在算法执行过程中，根据偏序关系，将算法中新产生的个体与 DA 中个体逐一进行比较，如果新个体劣于 DA 中某个个体，那么在本次迭代中不考虑该新个体。在采用精英保留策略之后，利用最近被淘汰的个体 j 对 DA 进行更新操作。基于偏序关系，如果 DA 中存在优于个体 j 的个体，那么 DA 保持不变；如果个体 j 优于 DA 中的某些个体，那么先将这些个体从 DA 中去除，再将个体 j 加入 DA 中；如果个体 j 和 DA 中的所有个体一样好，那么先删除在 DA 中进入集合时间最长的个体，再将个体 j 加入。

4.1.5 质量函数

应用引力搜索算法求解多目标优化问题时，质量函数的定义是算法的一个关键因素。一般情况下，质量函数可以通过适应度函数来定义。在处理单目标优化问题时，任意两个目标函数值都可以比较大小，因此群体所对应的解集合是全序的。适应度函数值可直接定义为目标函数值。个体的目标函数值越优，其质量越大。但是在处理多目标优化问题时，这种方法不再适用。在多目标优化中，所有非支配个体之间无法进行比较，这些个体所对应的解集合实际上是一种偏序集。同时在求解多目标优化问题时，不仅要求算法得到的非劣前端与 Pareto 最优前端的距离尽可能的小，而且要求其尽可能均匀分布。在设计适应度函数时，要尽可能地满足这两个要求。采用多目标优化问题的处理方法，即非支配排序和拥挤距离排序，给出适应度函数的定义。首先考虑层级最高(rank = 1)的个体，拥挤距离最大的个体适应度函数值为 1，拥挤距离第二大的个体适应度函数值为 2，按此方法将该层级上所有个体的适应度函数进行赋值；然后考虑层级第二高(rank = 2)的个体，根据拥挤距离，对该层级上个体的适应度函数值进行设置。以此类推，直到群体中所有个体都有相应的适应度函数值。最后，将个体质量定义为其适应度函数值的倒数。

4.1.6 算法流程

本章提出的 NMGSA 算法主要由引力搜索算法中的基本计算(包括计算个体质量和引力以及加速度、速度和位置的更新)、非支配排序、拥挤距离计算、精英保留策略和新个体与被删除个体的比较等操作组成。其中引力搜索算法的基本计算是产生新个体时必需的操作;通过非支配排序和拥挤距离的计算,对每个个体的性能进行定量分析;利用精英保留策略,从算法当前得到的解集和已有的 Pareto 最优解集中产生新的 Pareto 最优解集,并采用该 Pareto 最优解集产生新的个体;通过新个体和被删除个体的比较操作防止退化现象的产生,并对被删除个体集合进行更新操作。整个算法的主要步骤如下所述:

(1) 设置算法控制参数:群体 $P(t)$规模,Pareto 最优解集 $P_{known}(t)$规模,最大迭代次数 T,系数 G_0 和 α。

(2) 令 $t=1$,利用拟蒙特卡罗模拟方法中的 Sobol 序列产生初始群体 $P(t)$,Pareto 最优解集 $P_{known}(t-1)$ 为空集。

(3) 对个体进行非支配排序和拥挤距离计算。

(4) 计算个体质量和所受的引力。

(5) 进行加速度、速度和位置的更新操作。

(6) 将算法新产生的个体与被删除个体集合 DA 中的个体进行比较。

(7) 利用精英保留策略产生新的 Pareto 最优解集 $P_{known}(t)$,并对集合 DA 进行更新。

(8) 令 $t=t+1$,若当前迭代次数达到最大迭代次数,则算法停止,输出结果;否则转步骤 3。

4.2 收敛性证明

采用智能优化算法求解多目标优化问题已有近 30 年的时间,但是相关的研究侧重于算法的设计和数值实验结果的比较,对算法的理论研究较少。直到近些年,算法的理论分析才得到一定的关注,出现了一些收敛性方面的理论成果。基于本章给出的求解多目标优化问题的引力搜索算法,给出相关的收敛性分析。

在讨论多目标优化问题时,PF_{true} 的界是一个比较重要的概念,下面的两个定理给出了相关的研究结论。

定理 4.1[103,109] 给定一个多目标优化问题和非空有限可行解集,至少存在一

个 Pareto 最优解。

定理 4.2[103, 109] 任何多目标优化问题的 Pareto 最优 PF_{true} 最多由无穷多个向量组成。

当迭代次数趋于无穷大时，算法产生的 Pareto 前端和真实的 Pareto 最优前端一致，则认为该算法是收敛的。但在实际计算过程中，受内存资源和计算时间的限制，算法在求解多目标优化问题时仅能得到有限个数的 Pareto 解集。只要能够保证算法得到的 Pareto 解集是真实 Pareto 最优解集的子集，也可以认为算法是收敛的[102]。在此情形下，可以给出一个假设结论。

假设 4.1 在计算过程中，真实的 Pareto 最优解集是含有元素足够多的有界集。

定义 4.6[103, 109] 设 S 为 R^n 上的某个非空有界集，对任意的 $\delta > 0$，$N(\delta)$ 表示直径最大为 δ，可以覆盖 S 的集的最少个数。如果极限

$$\lim_{\delta \to 0} \frac{\ln N(\delta)}{\ln (1/\delta)} \tag{4.5}$$

存在，则称其为 S 的计盒维数。

定理 4.3[103, 109] 给定一个具有 m 个目标的优化问题和其最优边界 PF_{true}，如果 PF_{true} 是有界的，那么它是一个计盒维数不超过 $m-1$ 的集合。

算法的收敛过程可以通过 $P_{known}(t)$ 不断逼近 P_{true} 来实现，以下给出相关定义和定理。

定义 4.7[103, 109] 给定一个多目标优化问题，和其进化群体 Q_A 和 Q_B，定义 Q_A 和 Q_B 的关系为 $Q_A \geqslant Q_B$，如果满足条件：$\forall x \in Q_A$，$\nexists\ y \in Q_B$，使 $y > x$。

定理 4.4[103, 109] 给定多目标优化问题和多目标进化算法，如果满足条件：

(1) Pareto 最优边界的计盒维数不大于 $r-1$，r 为目标数；

(2) $P_{known}(0)$，$P_{known}(1)$，…，是单调的，即

$$\forall t: P_{known}(t+1) \geqslant P_{known}(t) \tag{4.6}$$

则算法以概率 1 收敛，即

$$prob(\lim_{t \to \infty}\{P_{known}(t) = P_{true}\}) = 1 \tag{4.7}$$

其中，$prob()$表示概率，P_{true} 为全局 Pareto 最优解集。

基于定理 4.4，给出求解多目标优化问题引力搜索算法的收敛性证明。根据前面的分析，给出了假设 4.1 的结论，并结合定理 4.3，定理 4.4 中的第一个条件能够得到满足。

欲证明算法产生的群体次序单调的，须要求算法的适应度函数和选择算法均为单调的。从适应度函数的定义可以看出，非劣层级不同的个体，层级较高的个体适应度函数值较小；同一层级上的个体，拥挤距离较大的个体适应度函数值较小。因此，这种基于非劣排序和拥挤距离比较设置适应度取值的方法具有单调性。

在求解多目标优化问题时，引力搜索算法采用精英保留策略。$P_{known}(t)$是在 $P_{known}(t-1)\cup P(t)$ 的基础上形成的，并且通过 $P(t)$和集合 DA 的比较操作，防止退化现象的产生。因此，第 t 代的 Pareto 解集至少不会出现劣于第 $t-1$ 代的结果。如果 $P(t)$中含有更好的非劣解，那么第 t 代的 $P_{known}(t)$解集将优于第 $t-1$ 代的 $P_{known}(t-1)$ 解集。因此，这种选择方法能满足单调性的要求。

综合上述证明过程，求解多目标优化问题的引力搜索算法满足定理 4.4 提出的两个条件，算法的收敛性得证。

必须指出的是，在保证求解多目标优化问题算法收敛的同时，也要求算法得到的 Pareto 解集尽可能均匀分布。但目前还没有对有关 Pareto 解集分布情况的理论研究，主要是通过实验结果进行分析。

4.3 数值实验

为测试本章算法 NMGSA 的性能，采用多个标准的多目标优化测试问题进行实验。测试函数包括 SCH、FON、DEB、VIN 和 4 个 ZDT(ZDT1、ZDT2、ZDT3 和 ZDT4)问题，这些函数都是目前国际上多目标优化领域中被广泛采用的测试函数[101-103,110]。前 4 个函数 SCH、FON、DEB 和 VIN 分别由 Schaffer、Fonseca、Deb 和 Vinnet 提出，后 4 个 ZDT 问题由 Zitzler 等人提出。这些测试问题最显著的特点是能够比较有效地测试一种多目标优化算法在处理多目标优化问题时，解群体能否收敛到 Pareto 最优前端以及能否保证分布的均匀性。前 4 个测试问题中，决策变量的数目最多为 3 个，目标函数个数最多为 3 个；后四个测试问题中，ZDT1、ZDT2 和 ZDT3 决策变量的数目为 30，ZDT4 决策变量的数目为 10，目标函数个数都为 2。表 4.1 是这些测试函数的数学描述。

表 4.1　多目标优化测试函数

问题	维数 n	变量取值范围	目标函数定义(最小化)
SCH	1	$[-10^3, 10^3]$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$
FON	3	$[-4, 4]$	$f_1(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right)$ $f_2(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right)$
DEB	2	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = (1+10x_2)\left(1-\left(\frac{x_1}{1+10x_2}\right)^2 - \frac{x_1}{1+10x_2}\sin(8\pi x_1)\right)$
VIN	2	$[-4, 4]$	$f_1(x) = \frac{(x_1-2)^2}{2} + \frac{(x_2+1)^2}{13} + 3$ $f_2(x) = \frac{(x_1+x_2-3)^2}{36} + \frac{(-x_1+x_2+2)^2}{8} - 17$ $f_3(x) = \frac{(x_1+2x_2-1)^2}{175} + \frac{(2x_2-x_1)^2}{17} - 13$
ZDT1	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)(1-\sqrt{x_1/g(x)})$ 其中 $g(x) = 1 + 9\sum_{i=2}^{n} x_1/(n-1)$
ZDT2	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)(1-(x_1/g(x))^2)$ 其中 $g(x) = 1 + 9\sum_{i=2}^{n} x_1/(n-1)$
ZDT3	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left(1-\sqrt{x_1/g(x)} - \frac{x_1}{g(x)}\sin(10\pi x_1)\right)$ 其中 $g(x) = 1 + 9\sum_{i=2}^{n} x_1/(n-1)$
ZDT4	10	$x_1 \in [0, 1]$ $x_i \in [0, 1]$ $(i = 2, 3, \cdots, n)$	$f_1(x) = x_1$ $f_2(x) = g(x)(1-\sqrt{x_1/g(x)})$ 其中 $g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))$

为求解多目标优化问题，很多多目标进化算法相继被提出，NSGA-Ⅱ是具有代表性的求解多目标优化的进化算法之一[106]，本章将其作为对比算法。在下面的实验中，NMGSA 的参数设置为：群体规模为 100，最优解集规模为 100，$G_0=30$，$\alpha=50$。NSGA-Ⅱ采用文献[106]中推荐的参数设置：群体规模为 100，最优解集规模为 100，交叉概率为 0.9，变异概率为 $1/n$，n 为决策变量个数。对于多目标进化算法，目前还很难给出明确的最优停止准则，大多数文献采用的方法是设定最大迭代次数或函数评价次数作为算法的终止条件。本章也采用这种方法，对这两种算法和八个测试函数，当函数评价次数达到 25 000 次时，算法就停止运行。这里，最大迭代次数相当于是 250 次。

在多目标优化问题中，一方面，对于同一个多目标优化问题而言，不同的多目标优化方法的计算结果往往并不相同；另一方面，多目标优化方法最终得到的结果是一个解集。因此，一个重要的多目标优化的研究内容就是不同求解方法之间优化性能的比较。

通常情况下，在分析多目标优化算法的性能时，至少希望算法在以下两个指标方面具有比较好的结果。

（1）算法最终得到的 PF_{known} 和问题真实的 PF_{true} 之间的距离应尽可能的短；

（2）算法最后求得的 Pareto 最优解点应具有较好的分布性，这些点在 Pareto 前端曲线或曲面上应尽量均匀分布。

本章采用两种常见的评价指标，即世代距离和空间度量指标，分别对算法的收敛性和分布程度进行评价[101-103]。

定义 4.8 世代距离（Generational Distance，GD）用来表示算法得到的 PF_{known} 和问题真正的 PF_{true} 之间间隔的距离，其计算方法如下：

$$GD \triangleq \frac{(\sum_{i=1}^{k} d_i^p)^p}{k} \tag{4.8}$$

其中，k 表示 PF_{known} 中向量的个数，通常 $p=2$，d_i 表示在目标空间中 PF_{known} 中第 i 个向量与 PF_{true} 中最近向量之间的欧氏距离。该指标值越小，说明算法得到的解的收敛性越好，越接近真实的 Pareto 前端。

定义 4.9 空间度量指标（Spacing，S）用于衡量 PF_{known} 中向量分布的均匀程度，其计算公式如下：

$$S \triangleq \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(\bar{d}-d_i)^2} \tag{4.9}$$

其中，$d_i = \min_j(|f_1^i(x)-f_1^j(x)|+|f_2^i(x)-f_2^j(x)|+\cdots+|f_m^i(x)-f_m^j(x)|)$，$m$ 表示目标函数的个数，$j \neq i$，i，$j=1, 2, \cdots, k$，k 表示 PF_{known} 中向量的个数，$\bar{d}$是所有 d_i 的平均值。如果 S 值越小，那么说明算法得到的非劣解集在目标空间的分布性越好。

一般认为，这两个指标中第一个指标是最重要的，因为多目标优化的目的首先是找到一组与真实 Pareto 前端的距离最近的非劣解[102]。在要求算法所获得的解尽量收敛之外，也希望解能在目标空间中尽量分布均匀。在实验中，对每个测试函数独立运行 30 次，考查算法得到的收敛性指标 GD 和分布性指标 S，分别统计这两种指标的最大值、最小值、平均值和标准差。实验结果如表 4.2 和 4.3 所示。

表 4.2　收敛性指标 GD 的测试结果

问　题	算　法	最大值	最小值	平均值	标准差
SCH	NSGA-Ⅱ	10.958 8	0.251 4	1.346 9	2.846 5
	NMGSA	0.348 8	0.328 7	0.338 4	0.005 2
FON	NSGA-Ⅱ	0.085 2	0.075 0	0.080 7	0.002 1
	NMGSA	0.083 2	0.070 3	0.077 8	0.003 1
DEB	NSGA-Ⅱ	0.139 7	0.110 9	0.118 0	0.006 2
	NMGSA	0.128 9	0.076 4	0.104 4	0.014 7
VIN	NSGA-Ⅱ	0.418 6	0.380 8	0.399 7	0.009 3
	NMGSA	0.404 4	0.377 7	0.390 7	0.007 4
ZDT1	NSGA-Ⅱ	0.129 8	0.103 0	0.119 2	0.006 8
	NMGSA	0.119 7	0.077 6	0.088 5	0.011 8
ZDT2	NSGA-Ⅱ	0.187 0	0.109 4	0.137 1	0.019 3
	NMGSA	0.132 1	0.039 5	0.076 9	0.015 5
ZDT3	NSGA-Ⅱ	0.204 8	0.154 5	0.178 5	0.012 3
	NMGSA	0.185 2	0.107 6	0.128 5	0.021 4
ZDT4	NSGA-Ⅱ	0.507 0	0.204 1	0.309 3	0.074 2
	NMGSA	0.108 7	0.071 6	0.081 0	0.005 9

表 4.3　均匀性指标 S 的测试结果

问　题	算　法	最大值	最小值	平均值	标准差
SCH	NSGA-Ⅱ	0.056 9	0.010 7	0.029 8	0.008 5
	NMGSA	0.044 1	0.025 8	0.035 0	0.004 2
FON	NSGA-Ⅱ	0.008 1	0.005 8	0.006 8	5.883 5e－004
	NMGSA	0.011 6	0.007 8	0.009 2	7.807 9e－004
DEB	NSGA-Ⅱ	0.011 0	0.004 8	0.007 0	0.001 1
	NMGSA	0.030 5	0.009 1	0.017 7	0.005 4
VIN	NSGA-Ⅱ	0.066 7	0.037 9	0.050 2	0.006 4
	NMGSA	0.082 1	0.060 8	0.068 6	0.006 0
ZDT1	NSGA-Ⅱ	0.013 3	0.005 9	0.008 2	0.001 6
	NMGSA	0.023 8	0.006 4	0.010 2	0.004 2
ZDT2	NSGA-Ⅱ	0.032 5	5.774 1e－007	0.006 7	0.006 4
	NMGSA	0.052 2	0.008 3	0.022 3	0.011 2
ZDT3	NSGA-Ⅱ	0.016 2	0.005 0	0.007 6	0.002 3
	NMGSA	0.050 7	0.007 2	0.015 7	0.009 9
ZDT4	NSGA-Ⅱ	0.013 4	0.000 5	0.009 2	0.002 9
	NMGSA	0.015 8	0.005 1	0.008 0	0.002 7

从表 4.2 可以看出，在考虑收敛性指标 GD 的情况下，与 NSGA-Ⅱ比较而言，NMGSA 的优势明显，特别是在求解 ZDT1、ZDT2、ZDT3 和 ZDT4 这 4 个较难优化函数时，效果十分显著。当考虑均匀性指标 S 时，从表 4.3 可以发现，NMGSA 和 NSGA-Ⅱ这两种算法得到的 S 值最大都不超过 0.09，表明这两种算法得到的非劣解集在目标空间具有较好的分布性，进一步比较发现，NSGA-Ⅱ的 S 值稍低于 NMGSA 的 S 值。根据前面对多目标优化评价指标重要程度的分析可知，首先要保证算法得到的解的收敛性，然后考虑解分布的均匀性。综合 GD 和 S 这两个指标，可以认为 NMGSA 在求解这些多目标优化问题时比 NSGA-Ⅱ更优越。

为较直观地说明 NMGSA 在求解多目标优化问题的性能，图 4.4 至 4.11 给出了对这些测试函数的仿真结果示意图。图中从上到下依次为 SCH、FON、DEB、VIN、ZDT1、ZDT2、ZDT3 和 ZDT4 的测试结果。

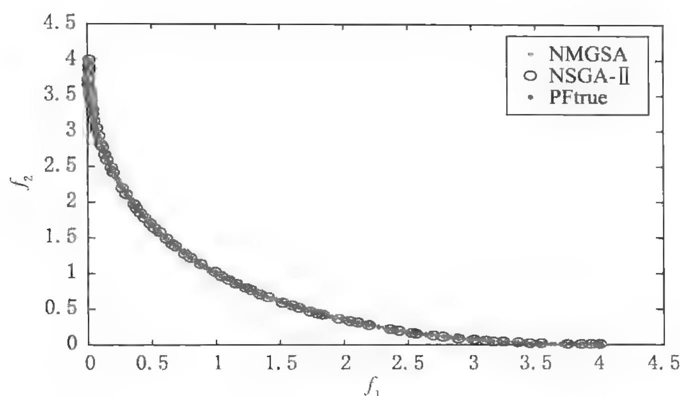


图 4.4　SCH 问题的 Pareto 前端和两种算法得到的 Pareto 前端

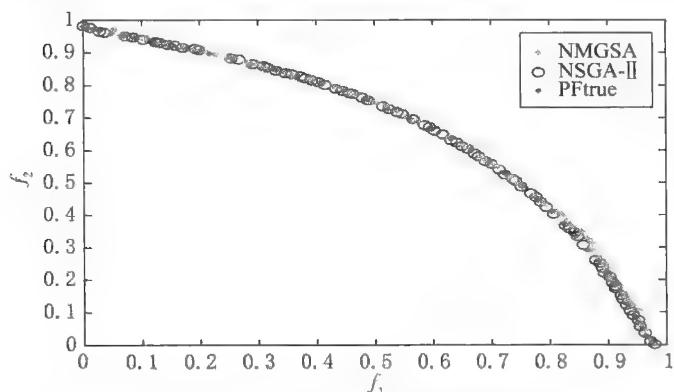


图 4.5　FON 问题的 Pareto 前端和两种算法得到的 Pareto 前端

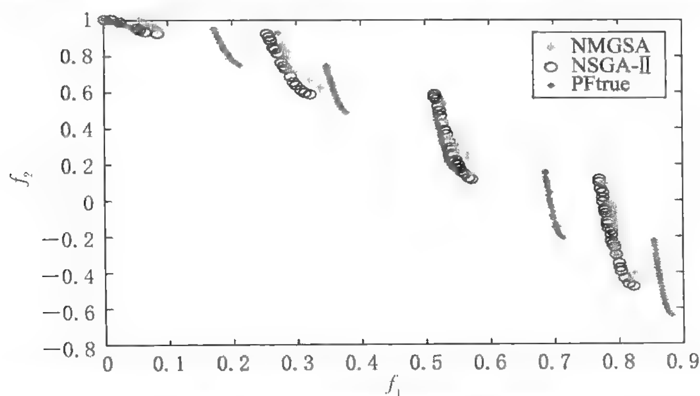


图 4.6　DEB 问题的 Pareto 前端和两种算法得到的 Pareto 前端

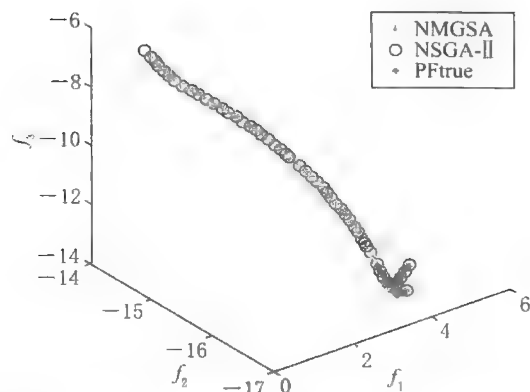


图 4.7　VIN 问题的 Pareto 前端和两种算法得到的 Pareto 前端

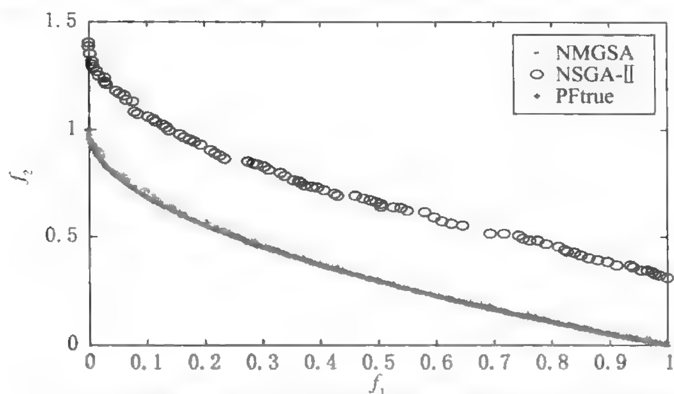


图 4.8　ZDT1 问题的 Pareto 前端和两种算法得到的 Pareto 前端

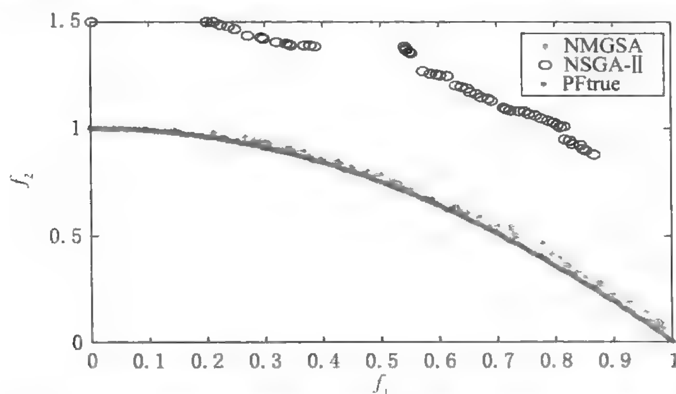


图 4.9　ZDT2 问题的 Pareto 前端和两种算法得到的 Pareto 前端

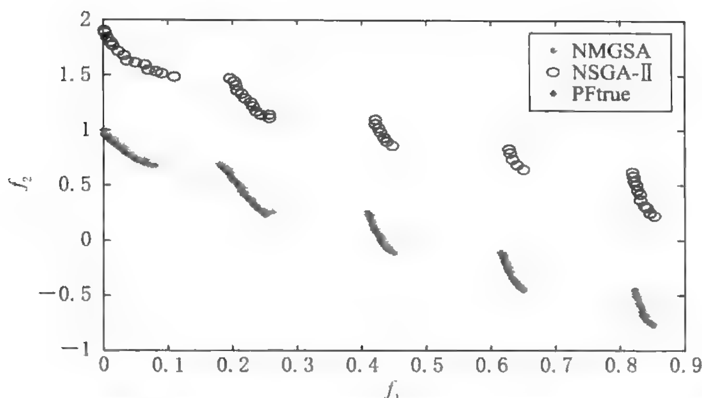


图 4.10　ZDT3 问题的 Pareto 前端和两种算法得到的 Pareto 前端

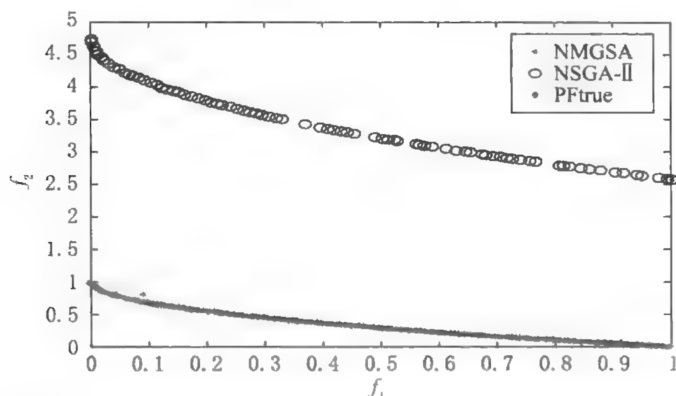


图 4.11　ZDT4 问题的 Pareto 前端和两种算法得到的 Pareto 前端

从这些图中可以看出，对于 SCH、FON 和 VIN 这 3 个问题，NMGSA 和 NSGA-Ⅱ这两种算法都取得的不错的结果，解不仅可以收敛到真实的 Pareto 前端，而且呈均匀分布状态。对于 DEB 问题，其 Pareto 前端为 6 个不连续的片段，这两种算法获得的结果都不甚理想，仅有少量解收敛到部分 Pareto 前端上。对于 ZDT1 问题，NMGSA 所获得的解不仅收敛性非常好，而且在目标空间均匀分布，而 NSGA-Ⅱ没有得到收敛到 Pareto 前端的解。对于 ZDT2 问题，NMGSA 获得的解非常接近真实的 Pareto 前端，而且解的分布均匀性较好，而 NSGA-Ⅱ得到的解与真实的 Pareto 前端距离较远。对于 ZDT3 问题，其 Pareto 前端为 5 个不连续的片段，NSGA-Ⅱ所获得的解没有收敛到任何一个有效区间上，而 NMGSA 得到的结果不仅

全部收敛到 Pareto 前端上，而且具有较好的分布均匀性。问题 ZDT4 的结果也同样表明了 NMGSA 的优越性。

为分析算法的优化速度，文献[102]和[111]定义了一个用于评价收敛速度的参数，称为进展度量，并用 RP 表示。计算公式如下：

$$RP = \ln\left(\sqrt{\frac{G_1}{G_t}}\right) \tag{4.10}$$

其中，G_1 表示第 1 次迭代的世代距离，G_t 表示第 t 次迭代的世代距离。一般情况下，RP 值在优化过程中逐渐变大，最后趋向某一固定值。图 4.12 至图 4.19 为这两种算法在运行过程中得到的 RP 值随迭代次数变化的情况。

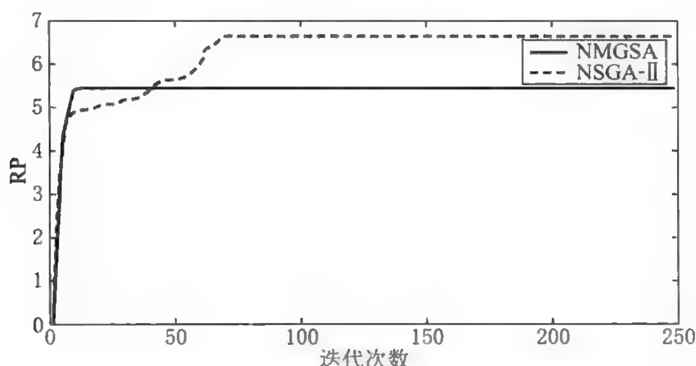


图 4.12　SCH 问题的进化过程中 RP 值

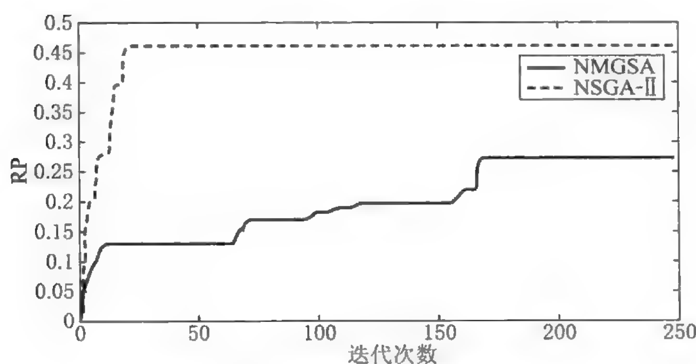


图 4.13　FON 问题的进化过程中 RP 值

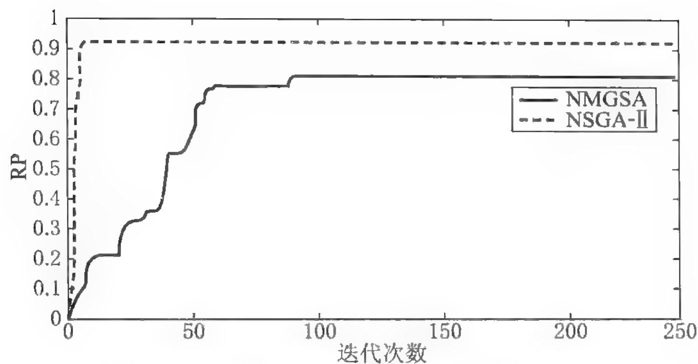


图 4.14 DEB 问题的进化过程中 RP 值

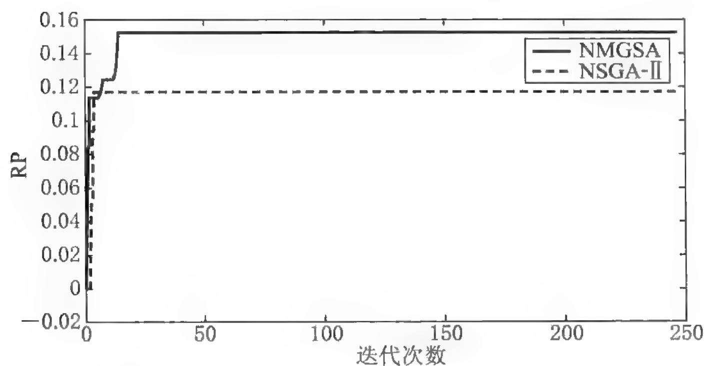


图 4.15 VIN 问题的进化过程中 RP 值

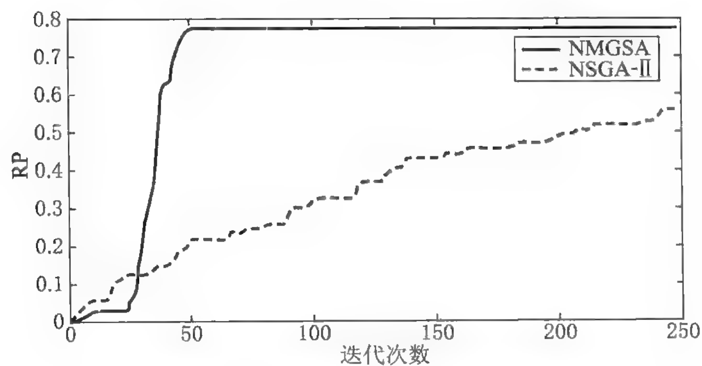


图 4.16 ZDT1 问题的进化过程中 RP 值

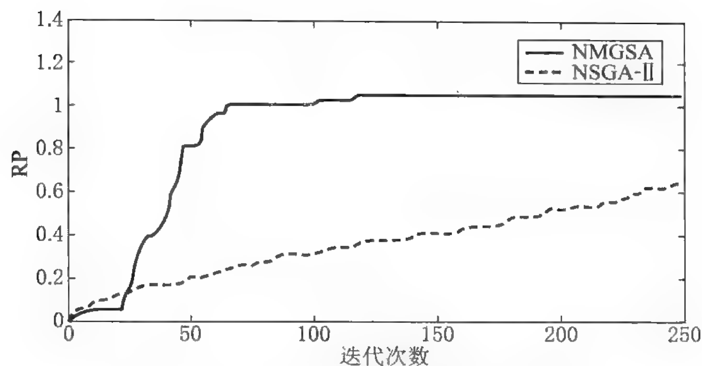


图 4.17　ZDT2 问题的进化过程中 RP 值

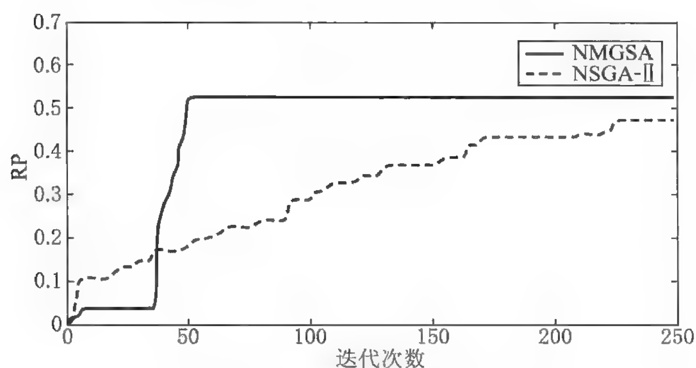


图 4.18　ZDT3 问题的进化过程中 RP 值

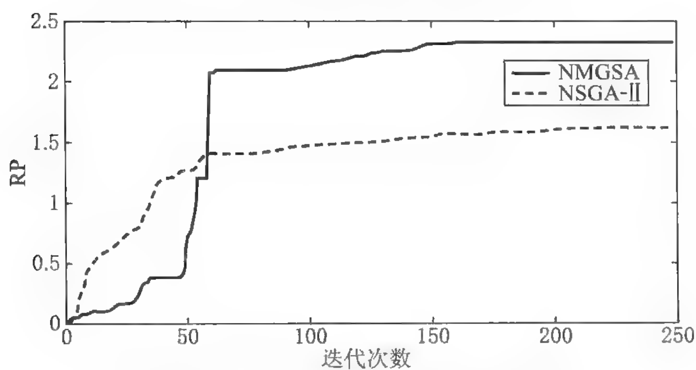


图 4.19　ZDT4 问题的进化过程中 RP 值

从这 8 个测试函数的 RP 值变化示意图可以看出，在 SCH、FON 和 DEB 这 3 个函数上，NSGA-Ⅱ的收敛速度要稍快于 NMGSA，但是在 VIN、ZDT1、ZDT2、ZDT3 和 ZDT4 这 5 个函数上，NMGSA 的收敛速度要快于 NSGA-Ⅱ，特别是在 ZDT1、ZDT2 和 ZDT4 这 3 个函数上，NMGSA 的优势较明显。就总体情况而言，与 NSGA-Ⅱ相比，NMGSA 具有更快的收敛速度。

第五章 引力搜索算法的若干应用

5.1 投资者偏好条件下概率准则投资组合问题

现代投资理论起源于 Markowitz 提出的证券投资组合理论，Markowitz 建立了均值——方差模型，提出了证券的组合投资是为了实现风险一定情况下的收益最大化或收益一定情况下的风险最小化[112-114]。但由于其苛刻的假设条件导致了该模型的应用性不强，许多研究者一直在努力对 Markowitz 的均值——方差模型进行拓展。基于投资者的投资心理，文献[114—122]提出了一种基于概率准则的投资组合模型，理论上克服了传统的证券收益率必须服从正态分布的局限性，同时对投资者进行证券投资选择时具有一定的指导意义。

由于目标函数是概率函数的形式，给求解带来一定的困难性。传统的以梯度为基础的优化方法通常要求目标函数连续或可导，而且处理不确定信息的能力较差。这些缺点使得传统优化算法在求解许多问题受到了限制。智能优化算法一般不要求目标函数连续性和可微性，甚至有时连有没有解析表达式都不要求，而且对计算中数据的不确定性也有很强的适应能力。目前将智能优化算法用于求解投资者偏好条件下概率准则投资组合问题的研究还很少，仅有遗传算法[122]。遗传算法根据适者生存，优胜劣汰的自然进化规则来进行优化计算，实现简单，鲁棒性好，但大量实验研究表明遗传算法会早熟收敛，易陷入局部极值。

针对投资者偏好条件下概率准则投资组合问题的特点，给出一种基于引力搜索算法的求解方法。在计算过程中，算法中寻优个体的质量对应优化的目标函数，个体的位置对应可能的组合证券选择。通过典型的实验和与遗传算法的比较，表明算法可行有效。

5.1.1 数学模型

假定投资者选择 D 种证券进行组合投资，令 $\xi=(\xi_1, \xi_2, \cdots, \xi_D)^T$，其中 ξ_i

表示第 i 种证券持有期的收益率($i=1, 2, \cdots, D$)；令 $x=(x_1, x_2, \cdots, x_D)^T$，其中 x_i 表示投资者用于第 i 种证券的投资比例系数，并假定市场不允许卖空，则 $\sum_{i=1}^{D} x_i = 1$, $x_i \geqslant 0$；组合证券的收益率为 $Y = x^T \xi = \sum_{i=1}^{D} x_i \xi_i$。

假定 R_0 表示组合证券的预期收益率，$P(Y \geqslant R_0)$ 表示实际证券组合收益不低于预期水平的概率值。概率准则下使证券投资组合达到预期收益率可能性最大的模型[122]可表示为：

$$\begin{aligned}&\operatorname*{Max}_{x}\ P(Y \geqslant R_0)\\&\text{s.t.}\ \begin{cases}\sum_{i=1}^{D} x_i = 1\\ x_i \geqslant 0 \quad (i=1, 2, \cdots, D)\end{cases}\end{aligned} \tag{5.1}$$

投资者在获得收益的同时也要承担一定的风险，投资者期望收益率低于某一灾难性水平的概率准则模型[122]如下所示：

$$\begin{aligned}&\operatorname*{Min}_{x}\ P(Y \leqslant R_1)\\&\text{s.t.}\ \begin{cases}\sum_{i=1}^{D} x_i = 1\\ x_i \geqslant 0 \quad (i=1, 2, \cdots, D)\end{cases}\end{aligned} \tag{5.2}$$

其中，R_1 表示灾难性水平或最小收益率，通常为负数。

证券投资组合的目标是为了实现最大可能的收益，但任何收益都伴随着风险。一般收益越大，风险也越大。不同的投资者有不同的偏好，有些投资者比较关注实现预期收益的把握性，有些投资者趋向保守，更注重灾难发生的概率。因此，引入投资者的偏好因子 λ，且 $\lambda \in [0, 1]$，投资者偏好条件下概率准则投资组合问题的模型[122]为

$$\begin{aligned}&\operatorname*{Max}_{x}\ \lambda P(Y \geqslant R_0) - (1-\lambda) P(Y \leqslant R_1)\\&\text{s.t.}\ \begin{cases}\sum_{i=1}^{D} x_i = 1\\ x_i \geqslant 0 \quad (i=1, 2, \cdots, D)\end{cases}\end{aligned} \tag{5.3}$$

模型(5.3)主要有两个特点：第一，对收益率服从任意概率分布的目标函数均可求解，突破了传统的证券收益率服从正态分布的假设；第二，考虑了投资者的偏

好，不同投资者根据自身的偏好程度选择相应的偏好因子进行优化计算，符合投资者的心理需求。

5.1.2 求解方法

采用引力搜索算法求解投资者偏好条件下概率准则投资组合问题，目标是使得基于投资者偏好的目标函数达到最大值。目标函数值与算法中寻优个体的质量对应；组合证券的投资比例系数和优化个体可行的位置对应；投资组合的初始选择和算法的初始解对应。基本引力搜索算法在求解优化问题时，对越界的处理方法为：若 $x < low$，则 $x = low$；若 $x > up$，则 $x = up$。其中 low 表示下界，up 表示上界，结合模型(5.3)的特点，本节中 $low = 0$，$up = 1$。若按原有的处理方法，所有越界的解都集中在边界上，即只取 0 或 1 两个值。从投资组合的规律来看，投资比例为 0 或 1 的情形比较少见，通常介于 0 到 1 之间。本节对处理方法做了如下改进：若 $x < low$ 或 $x > up$，则 $x = low + rand(up - low)$，其中 $rand$ 为 0 到 1 之间的随机数。通过这种操作，使得越界的解分散到可行空间内，不再聚集在边界上，符合投资组合中投资比例取值的特点，有利于改善算法的搜索效率。

本节提出的求解算法具体步骤如下：

第 1 步 初始化

考虑由 N 个物体组成的系统，N 个物体代表 N 种投资组合选择；每个物体定义在 D 维搜索空间中，维数 D 和待选择的证券总数相等；物体的位置代表优化问题的解。第 i 个物体的位置定义如下：

$$X_i = (x_i^1, \cdots, x_i^d, \cdots, x_i^D),\ i = 1, 2, \cdots, N \tag{5.4}$$

其中，x_i^d 表示第 i 个物体在第 d 维上的位置。

第 2 步 计算物体的质量

在第 t 次迭代时，物体的质量定义如下：

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{5.5}$$

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{N} q_j(t)} \tag{5.6}$$

其中，$M_i(t)$和 $fit_i(t)$分别表示在第 t 次迭代时第 i 个物体的质量和适应度函数

值，$best(t)$和$worst(t)$表示在第t次迭代时所有物体中最优适应度函数值和最差适应度函数值，对最大化问题，其定义如下：

$$worst(t) = \min_{j \in \{1, 2, \cdots, N\}} fit_j(t) \tag{5.7}$$

$$best(t) = \max_{j \in \{1, 2, \cdots, N\}} fit_j(t) \tag{5.8}$$

第3步 计算万有引力

物体i和j之间的万有引力定义如下：

$$F_{ij}^d(t) = G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t) - x_i^d(t)) \quad (d = 1, 2, \cdots, D) \tag{5.9}$$

其中，$G(t)$表示在第t次迭代时万有引力常数，$G(t) = G_0 e^{-\alpha t/T}$，$G_0$和$\alpha$为常数，$T$表示最大迭代次数；$R_{ij}(t)$表示物体$i$和$j$之间欧氏距离；$\varepsilon$是一个很小的常数，防止分母为零。

第4步 计算合力

物体i所受的合力为

$$F_i^d(t) = \sum_{j \in kbest,\, j \neq i}^{N} rand_j F_{ij}^d(t) \tag{5.10}$$

其中，$rand_j$表示在[0, 1]之间服从均匀分布的一个随机变量，$kbest$表示个体质量按降序排在前k个的个体，并且k的取值随迭代次数线性减小，初值为N，终值为1。

第5步 计算加速度

根据Newton第二运动定律，在第t次迭代时物体i在第d维的加速度为

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{5.11}$$

第6步 更新速度和位置

物体i的速度和位置在第d维的更新方程为

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{5.12}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{5.13}$$

第7步 如果没有达到最大迭代次数且没有退化行为（即找到的都是相同的解），转第2步；否则，停止计算，输出当前的最优解。

5.1.3 仿真实验

在进行实验前，先讨论有关目标函数的计算和约束条件处理的问题。

对服从任意分布的收益率或没有解析表达式的目标函数，可以用 Monte Carlo 模拟的方法计算概率。Monte Carlo 方法，又称随机抽样或统计试验方法。基本思想是当所要求解的问题是某种事件出现的概率，或者是某个随机变量的期望值时，它们可以通过某种“试验”的方法，得到这种事件出现的频率，或者这个随机变量的平均值，并用它们作为问题的解。以 $P(Y \geqslant R)$ 为例，给出 Monte Carlo 模拟的具体计算过程[118]。

步骤 1 对于固定的 x，令 $m = 0$；

步骤 2 从收益率的概率分布函数中生成随机向量 ξ；

步骤 3 若 $Y = x^T\xi \geqslant R$，则令 $m = m+1$；

步骤 4 重复步骤 2 和步骤 3，共 n 次；

步骤 5 由大数定律知，$P(Y \geqslant R) \approx \dfrac{m}{n}$。

模型(5.3)为约束优化问题，对约束条件的处理一般利用罚函数的方法将约束优化转化为无约束优化。本节针对该优化问题的特点，采用一种简单的处理方法。首先根据算法产生一组解 $x_1, x_2, \cdots, x_n (0 \leqslant x_i \leqslant 1)$，再进行归一化处理，即

$$x_i' = x_i / \sum_{i=1}^{n} x_i \tag{5.14}$$

在满足约束条件的要求后，进而直接考虑对目标函数的优化。

采用文献[122]中的典型算例进行数值计算。假设证券的收益率 ξ 服从混合正态分布，概率密度函数为 $f(x)$，且 $f(x) = \alpha f_1(x) + (1-\alpha) f_2(x)$，其中，$f_1(x)$ 是正态分布 $N(u_1, Q_1)$ 的概率密度函数，$f_2(x)$ 是正态分布 $N(u_2, Q_2)$ 的概率密度函数，常数 $\alpha \in [0, 1]$，实验中 α 取 0.3。其他数据如下：

$$u_1 = (0.0230, 0.0763, 0.3383, 0.7617, 0.4117, -0.2637)$$

$$Q_1 = \begin{pmatrix} 1.6967 & 0.5052 & -0.4144 & -0.9764 & 0.1327 & 0.5611 \\ 0.5052 & 1.7830 & -0.2389 & -0.0043 & 0.5711 & -0.0165 \\ -0.4144 & -0.2389 & 7.4756 & 0.4553 & -0.5256 & -1.0731 \\ -0.9764 & -0.0043 & 0.4553 & 4.2709 & 0.0454 & -0.3337 \\ 0.1327 & 0.5711 & -0.5256 & 0.0454 & 5.5701 & -1.3716 \\ 0.5611 & -0.0165 & -1.0731 & -0.3337 & -1.3716 & 2.2662 \end{pmatrix}$$

$u_2 = (0.0796, 0.3671, 0.3234, 0.1603, 0.0940, 0.5556)$

$$Q_2 = \begin{bmatrix} 2.1354 & 0.2848 & -0.2641 & 0.0597 & 0.2083 & 0.3237 \\ 0.2848 & 6.6053 & 0.1319 & 0.2235 & 0.4514 & 1.1873 \\ -0.2641 & 0.1319 & 6.9752 & -0.1091 & 1.3533 & -0.7517 \\ 0.0597 & 0.2235 & -0.1091 & 2.2668 & -0.4064 & 0.3043 \\ 0.2083 & 0.4514 & 1.3533 & -0.4064 & 3.3663 & -0.1656 \\ 0.3237 & 1.1873 & -0.7517 & 0.3043 & -0.1656 & 2.9665 \end{bmatrix}$$

证券的期望组合收益率 $R_0 = 0.18$，灾难性水平 $R_1 = -0.1$。

拟对不同偏好因子 λ 值进行测试，考虑三种常见的情况。第一种情况：投资者只关心实现预期收益的可能性，这类投资者往往只追求较高的收益回报而不考虑风险，属于激进型投资者，对应模型(5.3)中的 λ 取 1；第二种情况：投资者属于稳健型投资者，在关注收益的同时关注风险，此处 λ 取 0.5；第三种情况：投资者属于典型的风险厌恶者，不追求高额的回报，只关心风险发生的可能性，则取偏好因子 $\lambda = 0$。

采用引力搜索算法求解上述模型，并与文献[122]中遗传算法的求解结果进行比较。目前引力搜索算法的参数设定尚无理论依据，都是通过试验来确定，已公布的结果都是针对具体问题而言的。在本节中，引力搜索算法中的群体规模和最大迭代次数与文献[122]中遗传算法的设置相同，即群体规模为 20，最大迭代次数为 200。遗传算法其他参数采用文献[122]中的取值。引力搜索算法其他参数设置为 $G_0 = 100$ 和 $\alpha = 2$。两种算法的实验结果具体如下：

(1) 当 $\lambda = 1$ 时，

遗传算法的结果：

最大值 $P = 0.5561$，

投资比例系数 $x = [0.0006, 0.0311, 0.1811, 0.3995, 0.0435, 0.3442]$。

引力搜索算法的结果：

最大值 $P = 0.6280$，

投资比例系数 $x = [0.2465, 0.1200, 0.1800, 0.2199, 0.0108, 0.2229]$。

(2) 当 $\lambda = 0.5$ 时，

遗传算法的结果：

最大值 $P = 0.1140$，

投资比例系数 $x = [0.0133, 0.0591, 0.1374, 0.3105, 0.1400, 0.3396]$；

引力搜索算法的结果：

最大值 $P = 0.1640$，

投资比例系数 $x = [0.0721, 0.1524, 0.0785, 0.3313, 0.0603, 0.3054]$

(3) 当 $\lambda = 0$ 时，此时相当于求解模型(5.2)的最小值，

遗传算法的结果：

最小值 $P = 0.3218$，

投资比例系数 $x = [0.1104, 0.0643, 0.1210, 0.2676, 0.1486, 0.2881]$；

引力搜索算法的结果：

最小值 $P = 0.2480$，

投资比例系数 $x = [0.1780, 0.0491, 0.2014, 0.2400, 0.1854, 0.1463]$。

从上面的实验结果可知，引力搜索算法的求解结果优于遗传算法，为求解投资者偏好条件下概率准则投资组合问题提供了一个新的有竞争力的算法。在实验中发现，遗传算法和引力搜索算法，经过一定次数的迭代后，目标函数值趋向稳定，随着后续迭代次数的增加，目标函数值变化不大，算法具有较好的稳定性。

在实际应用中，投资者根据自身的偏好，选择相应的偏好因子，可利用本节提出的算法进行求解。遗传算法和引力搜索算法都属于智能优化算法，但两者的优化原理不同。遗传算法的优化思想源于自然选择原理和遗传机制，包括选择、交叉和变异三种算子；引力搜索算法基于万有引力定律进行寻优，采用速度和位置的计算模型，已经在一些复杂问题的求解中表现出良好的优化性能。

5.2 非线性极大极小问题

非线性极大极小问题是数学规划中一类典型的非光滑优化问题，要求在极大的条件下求目标函数的极小值。极大极小问题与曲线拟合、非线性方程组、非线性不等式组、多目标规划和对策论等有紧密的联系，同时在工程设计、电子电路规划等方面有着广泛的应用[123, 124]。因此，极大极小问题的研究有着重要的理论意义和实际价值。但由于目标函数不可微，给以梯度为基础的传统方法的求解带来了困难。近年来，智能优化算法的兴起为求解此类提供了新的思路。智

能优化算法不要求目标函数连续可微，也不需要梯度信息，使得算法具有较强的适应性。

为求解非线性极大极小问题，本节给出一种混沌引力搜索算法。混沌是非线性系统中较为普遍的一种现象，混沌运动具有遍历性、随机性、规律性等特点，能不重复地、均匀地遍历整个搜索空间。这些特性使得混沌优化能有效避免陷入局部极值，同时实验表明其具有较好的优化潜力[125-127]。将混沌优化引入到引力搜索算法中，对当前最优位置利用混沌搜索进行优化，对可能陷入局部最优的个体位置进行扰动，改变其寻优轨迹，跳出局部极值，提高搜索的效率和质量。

5.2.1 数学模型

考虑如下形式的极大极小问题：

$$\min \varphi(x) \tag{5.15}$$

其中，
$$\varphi(x) = \max\{f_i(x)\} \quad (1 \leqslant i \leqslant m) \tag{5.16}$$

上述模型中，$f_i(x)$为 $x \in R^n$ 的光滑实值函数；因目标函数 $\varphi(x)$在两个以上函数的交点处具有不连续的一阶导数，因此这类问题是一类不可微的非线性优化问题[128]。目前以梯度为基础的传统优化还不能有效求解。智能优化算法不要求目标函数连续可导，为求解此类问题提供了新的途径，为此，本节提出了一种混沌引力搜索算法的求解方法。

5.2.2 计算方法

基于混沌运动能在一定范围内按其自身规律不重复地遍历所有状态的特点，利用混沌优化进行局部搜索，改善优化性能。先将优化变量映射到混沌状态中，再把混沌运动的遍历范围逆映射到优化变量的搜索区间，利用混沌变量进行优化搜索。具体方法如下：假设 Lbest 表示当前最优位置，Up 表示上限，Low 表示下限，先将 Lbest 映射到[0, 1]区间，即令 $t=(Lbest-Low)/(Up-Low)$，利用 Logistic 映射 $t'=\mu t(1-t)$ 产生混沌变量 t'，再逆映射到搜索区间，即 $Lbest'=Low+t'(Up-Low)$，其中控制参数 $\mu=4$，系统完全处于混沌状态，有利于算法

跳出局部最优，重复上述步骤一定次数直到函数值不再改进。图 5.1 给出算法的伪代码。

```
1  Do random initialization;
2  Evaluate the fitness of the agents;
3  Calculate the mass and the total force;
4  Compute the acceleration and velocity;
5  Update the position;
6  Chaos optimization method is applied on the best position of the swarm;
7  Repeat Steps 2 to 6 until the stopping criterion is met
```

图 5.1　混沌引力搜索算法的伪代码

5.2.3　数值实验

为验证算法的性能，采用文献[128—130]中的几个典型算例进行测试，并与文献中的极大熵方法(又称凝聚函数法)和蚁群优化算法[55]的求解结果进行比较。极大熵方法具体流程参考文献[128]，混沌引力搜索算法和蚁群优化算法的群体规模和最大迭代次数分别设为 50 和 500，混沌引力搜索算法其他参数设置为 $G_0 = 100$ 和 $\alpha = 10$，蚁群优化算法的其他参数设置采用文献[55]中的推荐值。

例 5.1　$\min \varphi(x) = \min \max\{f_i(x)\},\ i = 1, 2, 3$

其中，
$$\begin{cases} f_1(x) = x_1^4 + x_2^2 \\ f_2(x) = (2 - x_1)^2 + (2 - x_2)^2 \\ f_3(x) = 2\exp(-x_1 + x_2) \end{cases}$$

已知问题的最优值为 $\varphi^* = \max\{f_1^*, f_2^*, f_3^*\} = 2$，其中，$f_1^* = f_2^* = f_3^* = 2$。凝聚函数法的最优结果为 $\varphi = \max\{f_1, f_2, f_3\} = 2.000\,000\,4$，其中 $f_1 = 1.999\,999\,8$，$f_2 = 2.000\,000\,4$，$f_3 = 1.999\,999\,3$；运行蚁群优化算法可得到最优解 $\varphi = \max\{f_1, f_2, f_3\} = 2$，其中，$f_1 = f_2 = f_3 = 2$；运行混沌引力搜索算法每次也可获得最优值 $\varphi = \max\{f_1, f_2, f_3\} = 2.000\,0$，其中 $f_1 = f_2 = f_3 = 2.000\,0$。

例 5.2 $\min\varphi(x)=\min\max\{f_i(x)\},\ i=1,2,3$

其中，
$$\begin{cases}f_1(x)=10+2x_1+2x_2-x_1^2+x_2^2-x_1x_2\\f_2(x)=2+x_1-x_2+2x_1^2-x_2^2+3x_1x_2\\f_3(x)=x_1^2+x_2^2\end{cases}$$

已知问题的最优解为 $\varphi^*=\max\{f_1^*,f_2^*,f_3^*\}=4.25$，其中，$f_1^*=f_2^*=f_3^*=4.25$。凝聚函数法获得的最优值为 $\varphi=\max\{f_1,f_2,f_3\}=4.250\,000\,3$，其中，$f_1=4.250\,000\,0$，$f_2=4.250\,000\,3$，$f_3=4.249\,999\,6$；反复运行蚁群优化算法都未能达到最优解，其最好结果为 $\varphi=\max\{f_1,f_2,f_3\}=4.490\,1$，其中 $f_1=4.490\,1$，$f_2=3.939\,8$，$f_3=4.187\,0$；运行混沌引力搜索算法均可得最优解 $\varphi=\max\{f_1,f_2,f_3\}=4.250\,0$，其中 $f_1=f_2=f_3=4.250\,0$。

例 5.3 $\min\varphi(x)=\min\max\{f_i(x)\},\ i=1,2$

其中，
$$\begin{cases}f_1(x)=|x_1+2x_2-7|\\f_2(x)=|2x_1+x_2-5|\end{cases}$$

已知问题的最优解为 $\varphi^*=\max\{f_1^*,f_2^*\}=0$，其中，$f_1^*=f_2^*=0$。多次运行蚁群优化算法均未能得到最优解，最好结果为 $\varphi=\max\{f_1,f_2\}=0.033\,6$，其中，$f_1=0.015\,9$，$f_2=0.033\,6$；运行混沌引力搜索算法每次都能得到最优解 $\varphi=\max\{f_1,f_2\}=0$，其中，$f_1=f_2=0$。

例 5.4 $\min\varphi(x)=\min\max\{f_i(x)\},\ i=1,2$

其中，
$$\begin{cases}f_1(x)=-x_1-x_2\\f_2(x)=-x_1-x_2+x_1^2+x_2^2-1\end{cases}$$

已知问题的最优解为 $\varphi^*=\max\{f_1^*,f_2^*\}=-1.414\,2$，其中，$f_1^*=f_2^*=-1.414\,2$。多次运行蚁群优化算法均未搜索到最优解，其最好结果为 $\varphi=\max\{f_1,f_2\}=-1.398\,1$，其中，$f_1=-1.406\,6$，$f_2=-1.398\,1$；运行混沌引力搜索算法每次都能得到最优解 $\varphi=\max\{f_1,f_2\}=-1.414\,2$，其中，$f_1=f_2=-1.414\,2$。

从以上典型算例的计算结果可以看出，混沌引力搜索算法比其他算法具有更强的获取全局最优解的能力。算法利用基于万有引力定律的寻优机制进行全局搜索，同时利用混沌优化对当前最优解区域进行集中探索，保证算法全局搜索和局部开发能力的平衡，避免算法陷入局部极值，具有较强的优化性能。

5.3 复杂系统可靠性优化问题

可靠性问题是系统设计、研究和运行过程中的一个重要因素，可靠性的高低直接影响系统的性能。若可靠性达不到相应的指标要求，系统越复杂，系统出故障的可能性越大，造成的损失也越大[131]。因此，如何对复杂系统可靠性进行优化已成为一个重要的研究课题。所谓复杂系统可靠性优化是指在满足一定的可靠性指标要求下使投资费用最小或在一定的资源约束条件下使系统的可靠度达到最大。由于其目标函数和约束条件都是非线性，而且目标函数通常都是具有多个局部极值的函数，这些给问题的求解带来了一定的困难。采用传统的优化算法求解往往达不到全局最优，解决此类问题难度较大。近年来，智能优化算法在求解很多复杂困难的优化问题中表现出良好的性能，已成为一个研究热点[132, 133]。目前用于复杂系统可靠性优化的智能算法有模拟退火算法[134]、遗传算法[135]、蚁群算法[136]等。这些算法在一定程度上可求解复杂系统可靠性优化问题，但算法都存在着各自的局限性，求解质量有待进一步改进。因此，探寻有效的求解方法颇为重要。

本节将序列二次规划引入基本引力搜索算法中，提出一种混合引力搜索算法，用于求解复杂系统可靠性优化问题。利用基本引力搜索算法进行全局搜索，并利用序列二次规划进行局部优化，实现全局探索和局部开发能力的平衡，避免基本引力搜索算法陷入局部极值。

5.3.1 数学模型

本节考虑一类典型的复杂系统可靠性优化问题[131-136]，对于一个给定的系统，在满足一定的可靠性约束条件下，通过为每一个子系统或部件选择合适的可靠性，使系统费用最小。

$$
\begin{aligned}
&\text{Min}\ C_s \\
&\text{s.t.}\quad \begin{cases} R_{i,\min} \leqslant R_i \leqslant 1 \quad (i=1,2,\cdots,n) \\ R_{s,\min} \leqslant R_s \leqslant 1 \end{cases}
\end{aligned}
\tag{5.17}
$$

其中，C_s 为系统费用；R_i 为部件可靠性；$R_{i,\min}$ 为部件可靠性下界；R_s 为系统可靠性；$R_{s,\min}$ 为系统可靠性下界。上述模型中，目标函数和约束函数均为非线性，而且目标函数具有多个局部极值，给寻找全局最优解带来了困难。

5.3.2 求解方法

现有的研究结果表明，基本引力搜索算法的全局探索能力较强，但局部开发能力较弱，需要嵌入其他的局部优化方法。序列二次规划（Sequential Quadratic Programming，SQP）最初由 Wilson 于 1963 年在其博士论文中提出，是解无约束优化问题的 Newton 法和 Quasi-Newton 法对约束优化问题的推广，具体是通过求解一系列二次规划的子问题逐步得到原问题的最优解[137]。目前 SQP 算法是非线性约束优化研究中一个十分活跃的领域。SQP 具有保持局部超 1 次收敛等特性，但一个不足之处是在优化有多个局部极值的目标函数时，对初始点的选取非常敏感，若选择不当会极大地影响算法的性能[138,139]。

基于引力搜索算法和序列二次规划各自的优缺点，提出一种混合引力搜索算法（Hybrid Gravitational Search Algorithm，HGSA）。算法首先利用引力搜索算法进行全局搜索，保存当前的最优位置，利用该最优位置作为 SQP 算法迭代的初始点，利用 SQP 进行局部搜索，有利于算法跳出局部极值，使算法全局搜索和局部开发的能力达到平衡。混合引力搜索算法的流程图如图 5.2 所示。

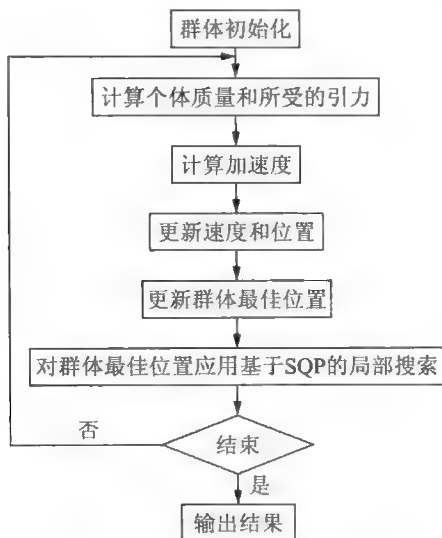


图 5.2 HGSA 算法流程图

5.3.3 实例计算

本节采用一种典型的复杂系统——非串联—并联系统[134-136]进行分析，系统结构图如图 5.3 所示。

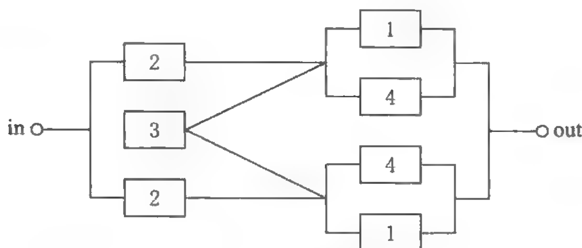


图 5.3 非串联—并联系统

系统可靠性 $R_s = 1 - R_3[(1-R_1)(1-R_4)]^2 - (1-R_3)\{(1-R_2[1-(1-R_1)(1-R_4)]\}^2$

系统费用 C_s 有两种形式：

(1)
$$C_s = 2\sum_{i=1}^{4} k_i R_i^{\alpha_i}$$

其中，$k_1 = 1$，$k_2 = 100$，$k_3 = 200$，$k_4 = 150$，$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.6$，$R_{i,\min} = 0.5$，$R_{s,\min} = 0.9$。

(2)
$$C_s = \sum_{i=1}^{4} k_i \left[\tan\left(\frac{\pi}{2}R_i\right)\right]^{\alpha_i}$$

其中，$k_1 = 25$，$k_2 = 25$，$k_3 = 50$，$k_4 = 37.5$，$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$，$R_{i,\min} = 0.5$，$R_{s,\min} = 0.99$。

在求解上述模型时，先将问题转换为无约束优化问题的形式。罚函数的形式一般有：和的形式、和的平方的形式、乘积形式等等。为方便起见，本节采用和的形式，这样在计算过程中可直接考虑对目标函数的优化，可行性的要求由罚函数的作用来强制其满足。

采用混合引力搜索算法 HGSA 进行求解，并与基本引力搜索算法 GSA，蚁群优化算法[55, 140]（Ant Colony Optimization，ACO）、微粒群算法[61]（Particle Swarm Optimization，PSO），人工蜂群算法[141, 142]（Artificial Bee Colony，ABC）的求解性能

进行比较。ACO 和 PSO 属于经典的智能优化算法，ABC 是最近提出的一种新的智能算法，这 3 种算法已成功用于求解很多复杂困难的优化问题，有关这 3 种算法的详细信息可参考文献[55，61，140—142]。

五种算法设置相同的群体规模和最大迭代次数，群体规模为 50，最大迭代次数为 50。在 ACO 算法中，信息启发因子 $\alpha = 1$，能见度因子 $\beta = 1$，轨迹的持久性参数 $\rho = 0.7$，信息素常数 $Q = 1$；在 PSO 算法中，学习因子 $c_1 = c_2 = 2$，惯性权重 w 从 0.9 线性递减到 0.2；在 ABC 算法中，引领蜂规模为 25，跟随蜂的规模为 25，侦察蜂规模为 1，阈值 $\lim it = 100$，采用轮盘赌的选择机制；GSA 算法和 HGSA 算法参数设置为 $G_0 = 100$ 和 $\alpha = 1$。每个算例独立运行 20 次并统计最优结果，具体结果如表 5.1 和 5.2 所示。

表 5.1　模型一的实验结果

Solution	ACO	PSO	ABC	GSA	HGSA
C_s	653.856 7	645.384 9	641.856 9	645.444 0	641.823 5
R_1	0.515 5	0.543 8	0.500 0	0.500 0	0.500 0
R_2	0.777 9	0.784 4	0.839 1	0.867 2	0.838 9
R_3	0.545 3	0.500 1	0.500 0	0.500 0	0.500 0
R_4	0.544 9	0.516 1	0.500 0	0.500 0	0.500 0
R_s	0.903 0	0.900 1	0.900 0	0.907 6	0.900 0

表 5.2　模型二的实验结果

Solution	ACO	PSO	ABC	GSA	HGSA
C_s	398.757 1	390.868 5	408.097 0	461.358 7	390.570 7
R_1	0.789 1	0.822 07	0.869 5	0.630 0	0.825 5
R_2	0.885 1	0.888 1	0.862 9	0.912 9	0.890 1
R_3	0.713 8	0.648 4	0.737 2	0.621 3	0.627 6
R_4	0.736 3	0.726 3	0.640 7	0.859 9	0.728 8
R_s	0.990 0	0.990 0	0.990 1	0.991 4	0.990 0

由表 5.1 和 5.2 可知，对模型一而言，HGSA 算法的最优结果小于 ABC 算法的

结果，远小于 GSA 算法、PSO 算法、ACO 算法的结果；对模型二而言，HGSA 算法的最好结果小于 PSO 算法的结果，比 GSA 算法、ABC 算法、ACO 算法的结果小得多，HGSA 算法表现出较高的优化精度。为进一步分析这几种算法的求解性能，图 5.4 和 5.5 给出了这 5 种算法的寻优曲线图。

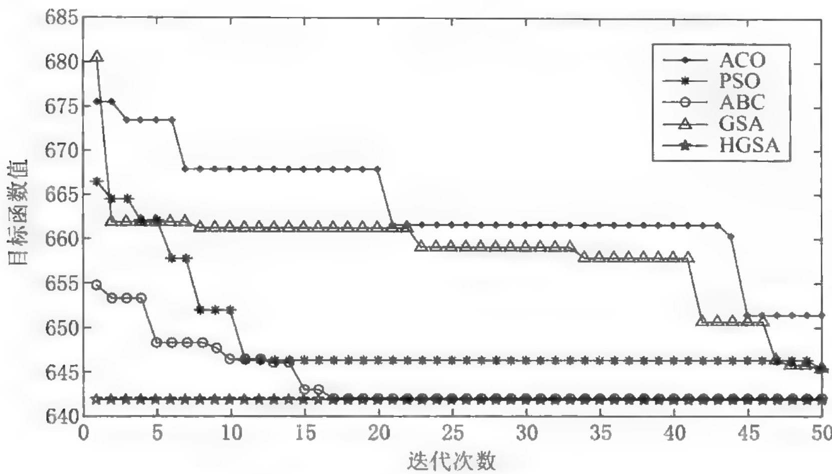


图 5.4　模型 1 的 5 种算法寻优曲线对比图

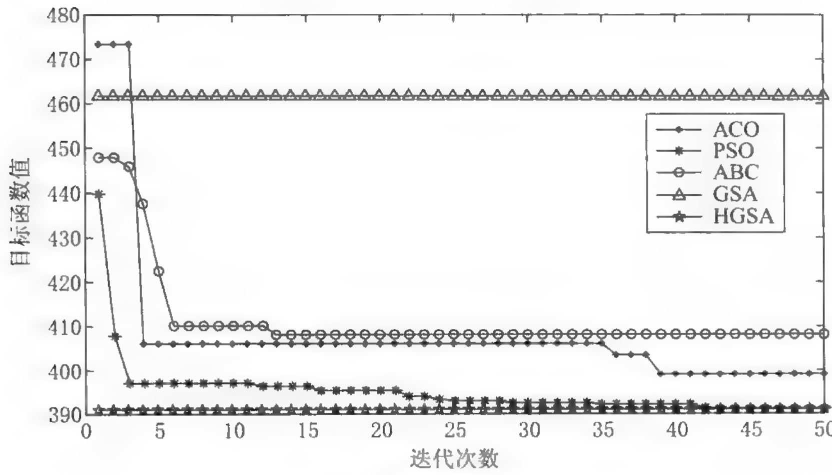


图 5.5　模型 2 的 5 种算法寻优曲线对比图

由图可知，在这5种算法中，HGSA算法具有最快的优化速度，算法只需很少的迭代次数就能达到最优并趋向稳定，算法具有较好的稳定性。HGSA算法在优化精度和寻优速度两方面均表现优异。分析原因，基本GSA算法全局搜索能力强而局部搜索能力弱，HGSA算法中嵌入了SQP算法，改善局部优化能力，能避免算法早熟收敛，实现全局探索和局部开发能力的平衡，提高算法优化性能。

5.4 多目标排水控制问题

5.4.1 问题描述

在中国南方的丘陵地区地表排水的过程中，因为对流与扩散的作用，土壤表层的可溶性氮和磷会迁移到地表径流中，造成氮和磷的流失。地表控制排水可以使排水中的悬移质与推移质容易沉淀，可以降低排水中的氮和磷的浓度，从而减少氮和磷的流失。同时地表排水会增加土壤入渗量，而为保证适合作物生长的土壤含水量与地下水位，要尽可能地减少入渗量。

以降低排水中的氮浓度、磷浓度和减少土壤入渗量为控制排水的目标，文献[143]建立了多目标排水控制模型。氮浓度、磷浓度与入渗量和排水控制时间的函数关系具体定义如下：

(1) 氮浓度的目标函数

根据各个时段的地表积水中氮的浓度，构造氮浓度和排水控制时间的函数关系：

$$\min X(t) = -5.22\ln(t) + 29.57 \tag{5.18}$$

其中，$X(t)$表示地表积水中的氮浓度，t表示排水控制时间，下同。

(2) 磷浓度的目标函数

根据各个时段的地表积水中磷的浓度，建立磷浓度和排水控制时间的函数关系：

$$\min Y(t) = -4.93\ln(t) + 27.62 \tag{5.19}$$

其中，$Y(t)$表示地表积水中的磷浓度。

(3) 土壤累计入渗量的目标函数

根据有关的降雨量和排水量，构造累计入渗量和排水控制时间的函数关系：

$$\min I(t) = 0.81\sqrt{t} + 0.14t \tag{5.20}$$

其中，$I(t)$表示累计入渗量。

从上述 3 个目标函数的表达式可以看出，氮浓度和磷浓度都是关于排水时间的递减函数，而累计入渗量是关于排水时间的递增函数。无法找到使得这 3 个目标同时达到最优的排水时间，但是可以找到 Pareto 最优的排水时间。

5.4.2 计算方法

采用第四章提出的求解多目标优化问题的引力搜索算法，优化多目标排水控制问题。算法的具体实现过程如表 5.3 所示。

表 5.3 多目标引力搜索算法的操作流程

步骤	操作
Step 1	设置参数，群体初始化。
Step 2	对群体进行非支配排序操作并计算拥挤距离。
Step 3	计算个体质量和所受的引力。
Step 4	计算个体的加速度，速度和位置。
Step 5	新个体与被删除个体的比较。
Step 6	采用精英保留策略产生新的 Pareto 最优解集。
Step 7	更新被删除个体的集合。
Step 8	若满足停机条件，则算法停止，输出结果；否则转 Step 2。

5.4.3 数值实验

采用我们提出的多目标引力搜索算法 NMGSA 求解上述模型，并与多目标微粒群算法[144]（Multiobjective Particle Swarm Optimization，MOPSO）的求解结果进行比较。NMGSA 的参数设置保持不变，MOPSO 的参数采用文献[144]中给出的推荐值，两种算法的最大函数评价次数为 25 000。在比较多目标优化算法的性能时，很多评价指标（如收敛性指标 GD）都需要所求解优化问题的真实的 Pareto 前端。而由于该多目标优化问题是实际问题，目前还无法预知其真实的 Pareto 前端。我们通过比较两种算法得到的 Pareto 前端来分析它们的优化性能。每种算法独立运行 20 次，并绘制这两种算法所得到的 Pareto 前端对比示意图，我们给出其中 3 种结果。

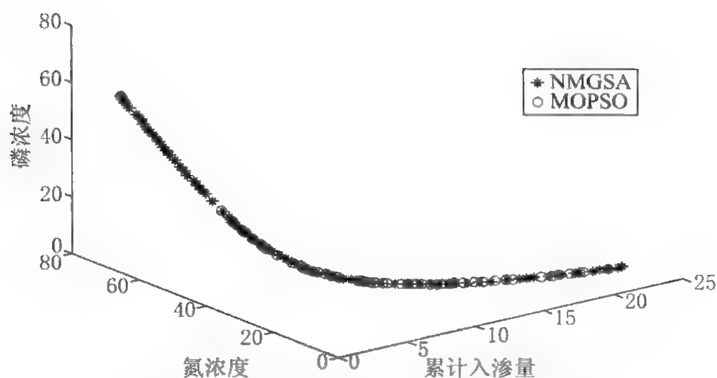


图 5.6　两种算法的 Pareto 前端示意图 1

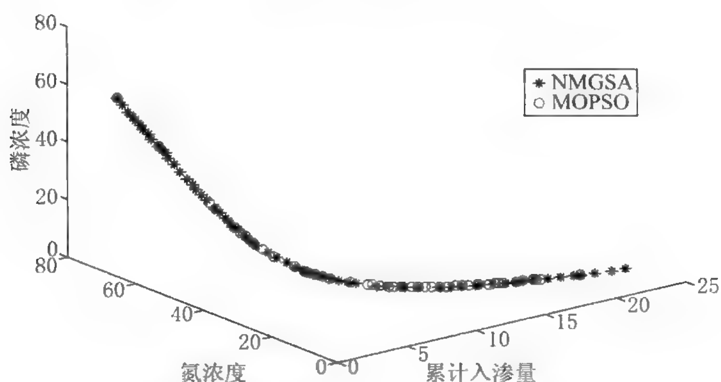


图 5.7　两种算法的 Pareto 前端示意图 2

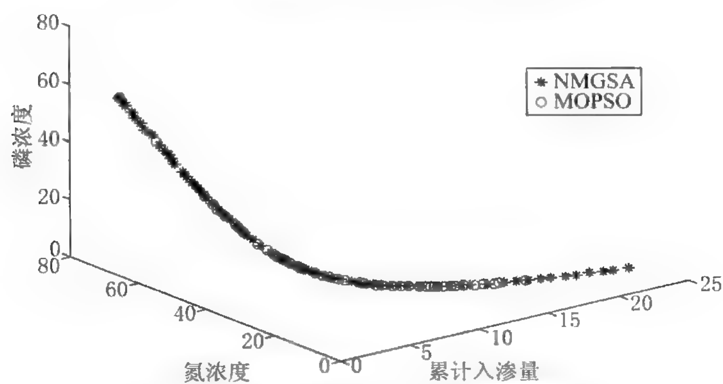


图 5.8　两种算法的 Pareto 前端示意图 3

从图 5.6 至 5.8 可以看出，NMGSA 算法得到的解不仅具有较好的收敛性，而且呈均匀分布状态。MOPSO 算法获得的非劣解的个数很少，而且分布不够均匀，部分区域出现重叠现象。一个多目标算法得到的非劣解越多、分布性越好，能为决策者提供更多更好的选择方案。在实际问题中，决策者可根据具体情况选择一个或多个合适的非劣解。此外，我们注意到，在 20 次的独立运行中，NMGSA 算法所获得的 Pareto 前端差别非常小，说明 NMGSA 在求解该问题上的优越性是稳定的。

第六章　离散优化的元胞引力搜索算法

考虑到引力搜索算法独特的优化机制，已经有少数研究者开始尝试利用引力搜索算法求解离散优化问题，例如文献[85]将算法用于求解流水线调度问题。从研究结果来看，引力搜索算法的全局优化性能较好，而局部搜索能力较弱。因此，需要嵌入一些专门的辅助方法，来提高算法的搜索性能。

元胞自动机由 Von Neuman 提出[115]，是一个时间、空间、状态都离散的动力学系统，已经成为以离散性为特点描述复杂行为的一种具有广阔发展前景的方法[116]。在元胞自动机中，重复简单的计算规则能导致复杂的系统行为，反复进行局部的交互作用能实现全局计算的目的。将元胞自动机用于智能优化算法正是基于这样的一个出发点，我们已经将元胞自动机和微粒群算法与蚁群算法相结合，并取得了较为满意的结果。在本章中，将给出这方面的研究成果[147,148]，一方面说明元胞自动机在改善智能优化算法求解性能方面的优越性，另一方面为元胞自动机和引力搜索算法的结合提供思路。在此基础上，给出一种元胞引力搜索算法，并以最小比率旅行商问题为例，验证所提出算法的优化性能。

6.1　元胞自动机在智能优化算法中的应用举例

6.1.1　元胞自动机原理

元胞自动机，又称为分子自动机、细胞自动机或点格自动机，是一种在时间、空间和状态上都离散的动力学模型。元胞自动机的主要原理是通过大量元胞在简单规则下的演化来模拟复杂现象。每一个散布在规则网格中的元胞只取有限的离散状态，并遵循同样的作用规则，根据确定的局部规则进行同步更新。通过大量元胞的相互作用来构成系统的演化[55,149]。

元胞自动机最基本的组成部分是元胞、元胞空间、邻居和规则[55,117]。简单地说，元胞自动机可视作一个元胞空间以及定义在该空间上的变换函数组成的，

如图 6.1 所示。

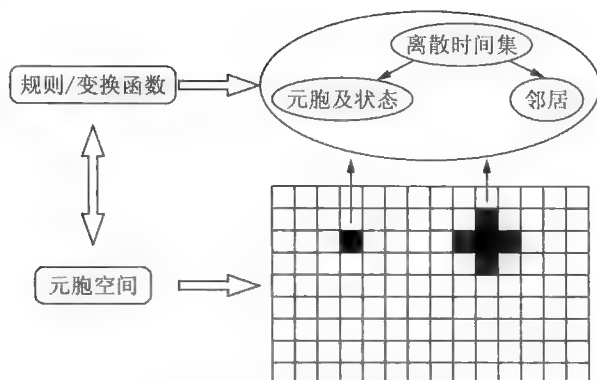


图 6.1　元胞自动机的组成

虽然元胞自动机有着比较宽松、甚至近乎模糊的组成条件，但是作为一个数学模型，元胞自动机有着严格的描述。下面从集合论的角度，给出元胞自动机的数学定义[55, 150]。

假设 d 表示空间维数，k 表示元胞的状态，并且在一个有限集合 S 中进行取值，r 表示元胞邻居的半径。Z 为整数集，代表一维空间，t 表示时间。

为叙述与理解的方便起见，可以在一维空间上考虑元胞自动机，即设定 $d=1$，那么整个元胞空间就是在一维空间上。将整数集 Z 上的状态集合 S 的分布，表示为 S^Z。

元胞自动机的演化就是在时间上的状态组合的动态变化，可以表示为：

$$F:\ S_t^Z \rightarrow S_{t+1}^Z \tag{6.1}$$

这个演化过程又是由各元胞的局部函数 f 决定的。这个局部函数 f 往往又被称为局部演化规则。

对一维空间而言，元胞及其邻居可以表示为 S^{2r+1}，局部函数则可以表示为：

$$f:\ S_t^{2r+1} \rightarrow S_{t+1} \tag{6.2}$$

对于局部演化规则 f 而言，函数的输入和输出集都是有限集合。本质上，它是一个有限参照表。例如，当 $r=1$ 时，f 的形式可以进行如下的定义：

[0, 0, 0]→0; [0, 0, 1]→0; [0, 1, 0]→1; [1, 0, 0]→0;

[0, 1, 1]→1; [1, 0, 1]→0; [1, 1, 0]→0; [1, 1, 1]→0。

对元胞空间中的每个元胞，分别施加上述形式的局部函数，则可以获得全局的演化：

$$F(c_{t+1}^{i}) = f(c_t^{i-r}, \cdots c_t^{i}, \cdots c_t^{i+r}) \tag{6.3}$$

其中，c_t^i 表示在 t 时刻和在位置 i 处的元胞的状态。至此，我们就获得了一种元胞自动机的模型。

标准元胞自动机可以用一个四元组表示：

$$A = (L_d, S, N, f) \tag{6.4}$$

其中，A 代表一个元胞自动机的系统；L 代表元胞空间，d 为一个正整数，代表元胞空间的维数；S 代表元胞有限且离散的状态集合；N 代表一个所有邻域中元胞的可能组合（包含中心元胞），即含有 n 个不同元胞状态的空间矢量，可表示为：

$$N = (S_1, S_2, \cdots, S_n) \tag{6.5}$$

这里，n 表示元胞邻居的个数；$s_i \in Z$（整数集合），$i \in \{1, 2, \cdots, n\}$；$f$ 表示一种局部转换函数。所有的元胞都位于 d 维空间内，其相应位置可以采用一个 d 元的整数矩阵 Z^d 进行确定。

6.1.2 元胞蚁群算法

(1) 算法设计

以非线性 0—1 规划为例，给出元胞蚁群优化算法的数学描述。非线性 0—1 规划的模型可以表示为：

$$\min f(x)$$
$$\text{s.t.} \begin{cases} g_i(x) \leqslant b & (i = 1, 2, \cdots, s) \\ h_j(x) = c & (j = 1, 2, \cdots, t) \\ x_k \in \{0, 1\} & (k = 1, 2, \cdots, n) \end{cases} \tag{6.6}$$

其中，$f(x)$ 为非线性函数；$g(x)$ 和 $h(x)$ 为线性函数或非线性函数；s 为不等式约束的个数；t 为等式约束的个数；x 为 n 维 0—1 变量。

定义 6.1 设集合 $C=(c_1, c_2, \cdots, c_i, \cdots, c_n)$，其中 $c_i \in \{0, 1\}$，C 中 c_i 任意取值的排序组合的集合为元胞空间，可表示为 $L=\{CellX=(c_1, c_2, \cdots, c_i, \cdots, c_n) \mid c_i \in \{0, 1\}\}$，每个组合 $CellX$ 为元胞。

定义 6.2 扩展 *Moore* 邻居类型：

$$N_{moore}=\{CellY \mid diff(CellY-CellX) \leqslant R, CellX, CellY \in L\} \tag{6.7}$$

其中，$diff(CellY-CellX) \leqslant R$ 为两个组合排序的差异，若无差异为 0，有差异时，最小为 2。R 为差异度，这里 R 取 2。

定义 6.3 元胞演化规则

依据元胞邻居的定义计算其邻居的目标解，比较元胞和其邻居的差异，选择最好的目标解。

定义 6.4 转移概率

$$p_{ij}=\frac{[\tau_j]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_k [\tau_k]^{\alpha}[\eta_{ik}]^{\beta}} \tag{6.8}$$

其中，τ_j 为轨迹强度；η_{ij} 为能见度，$\eta_{ij}=f_j-f_i$，f_j 和 f_i 分别表示第 j 个和第 i 个蚂蚁的目标函数值；α 为信息素强度的重要性；β 为启发式因子的重要性。

定义 6.5 信息素强度更新方程

$$\tau_j^{new}=\rho\tau_j^{old}+\sum_k \Delta\tau_j^k \tag{6.9}$$

其中，ρ 体现强度的持久性，$\Delta\tau_j^k$ 采用 Ant-Density 模型。

非线性 0 1 规划问题的元胞蚁群算法主要步骤如下：

Step 1 $nc \leftarrow 0$（nc 为迭代次数），各参数初始化。

Step 2 将各蚂蚁的初始出发点置于当前解集中，每个蚂蚁进行搜索，计算各蚂蚁的目标函数值。

Step 3 按元胞邻居的定义，在邻居范围内演化，记录最好的解。

Step 4 按信息素强度更新方程修改轨迹强度。

Step 5 令 $nc \leftarrow nc+1$，$\Delta\tau \leftarrow 0$。

Step 6 若 $nc<$ 预定的迭代次数且无退化行为（即找到的都是相同解），则转 Step 2；否则，转 Step 7。

Step 7 输出目前的最好解。

(2) 数值实验

为测试元胞蚁群算法的有效性，本节求解了一系列具有非唯一最优解的算例，并将其求解结果和多种算法的计算结果进行比较。算法参数设置如下，$\alpha=\beta=1$，$\rho=0.9$，$Q=10$，$ants=3$，$nc=20$。实验所用硬件为 Pentium 3.4 GHz，1 GRAM，软件为 Windows XP 和 Matlab。

首先将元胞蚁群算法与蚁群算法的全局寻优性能相比较，算例来自文献[151]和[152]，蚁群算法中的参数与元胞蚁群算法中的参数取值一致。

例 6.1[151]

$$\min x_2^2 - 2x_1x_2 - x_2$$
$$\text{s.t.} \begin{cases} x_1 + x_2 \leqslant 1 \\ x_2 - x_1 \leqslant 0 \\ x_1,\ x_2 \in \{0,\ 1\} \end{cases}$$

表 6.1　算例 1 的计算结果

算　法	蚁群算法	元胞蚁群算法
最优解	(0, 0)	(0, 0)
		(1, 0)

例 6.2[152]

$$\min 3x_1x_2x_4 + 2x_2x_3x_4 + 9x_2x_4 + 5x_1 - 2x_2 + 7x_4 - 17$$
$$\text{s.t.} \begin{cases} x_1x_2x_3x_4 + 2x_2x_3x_4 + x_2x_4 + 6x_1 - 2x_2 + 6x_4 = 6 \\ x_1x_2x_3x_4 + 2x_2x_3x_4 + x_2x_4 + 7x_1 - 2x_2 + 7x_4 = 7 \\ x_i \in \{0,\ 1\} \quad (i = 1,\ \cdots,\ 4) \end{cases}$$

表 6.2　算例 2 的计算结果

算　法	蚁群算法	元胞蚁群算法
最优解	(1, 0, 0, 0)	(1, 0, 0, 0)
		(1, 0, 1, 0)

从表 6.1 和表 6.2 可以看出，元胞蚁群算法改善了蚁群算法的搜索能力，具

有更强的全局寻优能力。为进一步测试算法全局优化的性能，采用文献[153—155]中的算例，并与文献中相应算法的性能进行比较，相关算法的含义及参数设置参考文献[153—155]。

例 6.3[153]

$$\min -9x_1+3x_3+3x_5+12x_1x_3+12x_1x_4+48x_2x_4+36x_2x_6+60x_4x_6$$

$$\text{s.t.}\quad \begin{cases} x_1+x_2=1 \\ x_3+x_4=1 \\ x_5+x_6=1 \\ x_i\in\{0,\ 1\}\quad (i=1,\ \cdots,\ 6) \end{cases}$$

表 6.3　算例 3 的计算结果

算　法	禁忌搜索算法[153]	元胞蚁群算法
最优解	(1, 0, 0, 1, 1, 0)	(0, 1, 1, 0, 1, 0)
		(1, 0, 1, 0, 0, 1)
		(1, 0, 0, 1, 1, 0)

例 6.4[154]

$$\min -3x_1+x_3+x_5+4x_1x_3+4x_1x_4+16x_2x_4+12x_2x_6+20x_4x_6$$

$$\text{s.t.}\quad \begin{cases} x_1+x_2=1 \\ x_3+x_4=1 \\ x_5+x_6=1 \\ x_i\in\{0,\ 1\}\quad (i=1,\ \cdots,\ 6) \end{cases}$$

表 6.4　算例 4 的计算结果

算　法	遗传算法[154]	元胞蚁群算法
最优解	(1, 0, 0, 1, 1, 0)	(1, 0, 0, 1, 1, 0)
		(0, 1, 1, 0, 1, 0)
		(1, 0, 1, 0, 0, 1)

例 6.5[155]

$$\min 3x_1^5+4x_1x_3+2x_2^3-7x_4x_5x_6+2x_4^2+5x_5^3+8x_6^2-4x_1x_2-3x_1x_3$$

$$\text{s.t.}\begin{cases}4x_1+3x_2+2x_3+3x_4+4x_5+6x_6\geqslant 14\\3x_1+2x_2+3x_3+4x_4+2x_5+3x_6\geqslant 10\\x_i\in\{0,1\}\quad(i=1,\cdots,6)\end{cases}$$

表 6.5　算例 5 的计算结果

算　法	乘子法[155]	约束松弛法[155]	元胞蚁群算法
最优解	(1, 1, 0, 1, 1, 0)	(0.999 9, 1.000 0, 0.000 0, 0.993 9, 0.004 0)	(1, 1, 0, 1, 1, 0)
			(0, 0, 1, 1, 1, 1)

目前对线性 0—1 规划问题还没有很有效的求解方法，如背包问题是典型的线性 0—1 规划，是经典的 *NP* 难题，当求解目标函数为非线性时，求解难度急剧加大。表 6.1 至表 6.5 的实验结果表明，元胞蚁群算法具有较强的搜索多个全局最优解的能力，算法与蚁群算法、禁忌搜索算法、遗传算法、乘子法、约束松弛法相比，可以得到更多的全局最优解。算法将元胞自动机原理和蚁群算法相融合，利用元胞及其邻居扩大蚁群搜索空间，增强算法的多样性搜索，提高获得多个最优解的概率；每只蚂蚁按演化规则的更新是同步进行的，具有并行性；将元胞演化规则和信息素更新规则相结合，更好的引导蚁群寻优，蚁群对解质量较好的区域进行集中搜索，提高优化速度。算法具有多样化搜索、集中搜索和并行计算等特点，随着优化过程的进行，同时向多个最优解进化。

该算法也具有一定的通用性，文献[153]提出的 Tabu Search Algorithm 和文献[154]提出的遗传算法只能求解一类特殊的二次 0—1 规划；文献[155]提出的乘子法只能求解目标函数为非线性、约束为线性的 0—1 规划问题，约束松弛法对约束为线性和非线性的非线性 0—1 规划均可求解，但用这两种方法处理后还需使用遗传算法进一步求解；而元胞蚁群算法对上述文献提出的模型均可有效求解，算法和目标函数的具体形式无关，可适用于各种非线性 0—1 规划问题。元胞蚁群算法拓宽了元胞自动机的应用范围，目前元胞自动机在优化方面的应用很少，将元胞自动机引入到蚁群算法中，为解决复杂优化问题提供了一个新的有效的方法。

为测试算法求解大规模问题的能力，采用文献[156]的模型，进一步测试本算法的性能。参数设置为 $\alpha=\beta=1$、$\rho=0.9$，$Q=10$，$ants=10$，$nc=50$。

例 6.6[154]

$$\max f(x)=\sum_{j=1}^{n}c_jx_j+\sum_{i=1}^{q}d_i\prod_{j\in N_i}x_j$$

$$\text{s.t.}\quad\begin{cases}\sum_{j=1}^{n}a_jx_j\leqslant b\\ x_j\in\{0,1\}\quad(j=1,2,\cdots,n)\end{cases}$$

测试问题的系数随机产生，$c_j\in[1,100]$，d_i，$a_j\in[1,50]$，$b\in[50,\sum_{j=1}^{n}a_j]$，$N_i(2\leqslant|N_i|\leqslant 6)$中的指标是在$\{1,2,\cdots,n\}$随机产生。每个实验独立运行 20 次，计算结果见表 6.6，其中，n 为 0—1 变量个数；q 为 $f(x)$中非线性项的个数；T 为 20 个测试问题的平均运行时间。

表 6.6　算例 6 的计算结果

n	T(s)			
	$q=20$	$q=30$	$q=40$	$q=50$
50	62.81	81.70	75.84	85.85
80	234.28	258.82	288.87	306.11
100	490.43	550.87	586.42	628.40

从表 6.6 的数据可以看出，本节提出的算法能在比较合理的时间内求解较大规模的问题。对较大规模非线性 0—1 规划问题的求解一直是一类很困难的问题，仍然缺乏有效的手段，本算法是解决此类较大规模模型的一种有力的求解方法。

6.1.3　元胞微粒群算法

(1) 算法设计

以多维背包问题为例，给出元胞微粒群算法的设计方案。多维背包问题的数学模型为：

$$\text{Max} f(x_1, x_2, \cdots, x_n) = \sum_{j=1}^{n} c_j x_j$$

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i & (i = 1, 2, \cdots, m) \\ x_j \in \{0, 1\} & (j = 1, 2, \cdots, n) \end{cases} \tag{6.10}$$

其中，n 为物品的编号，m 为资源的编号；c_j 为第 j 个物品的受益量；b_i 为第 i 种资源的预算；a_{ij} 为第 j 个物品占用第 i 种资源的量；x_j 为0—1决策变量（当物品 j 被选择时 $x_j = 1$，否则 $x_j = 0$）。

对于多维背包问题，其可行域集合为 n 维欧氏空间，可将其中的任一元素视为元胞，而所有元素构成元胞空间。结合多维背包问题，给出元胞微粒群算法的具体描述。

定义 6.6 设物品选择结果的集合 $C = (c_1, c_2, \cdots, c_i, \cdots, c_n)$，其中 $c_i \in \{0, 1\}$，$c_i = 1$ 表示选中物品 i，$c_i = 0$ 表示未选中物品 i。C 中 c_i 任意取值的排序组合的集合为元胞空间，可表示为 $L = \{CellX = (c_1, c_2, \cdots, c_i, \cdots, c_n) \mid c_i \in \{0, 1\}\}$，每个组合 $CellX$ 为元胞。

定义 6.7 *Moore* 邻居类型：

$$N_{moore} = \{CellY \mid diff(CellY - CellX) \leqslant r, CellX, CellY \in L\} \tag{6.11}$$

其中，$diff(CellY - CellX) \leqslant r$ 为两个组合排序的差异，若无差异为0，有差异时，最小为1。r 为差异度，这里 r 取1。

定义 6.8 微粒的位置和速度更新方程

设解空间的维数为 d，第 i 个微粒在进化的第 k 代时位置为 $x_i^k = (x_{i1}^k, x_{i2}^k, \cdots, x_{id}^k)^T$，微粒已搜索过的最好位置为 $p_i^k = (p_{i1}^k, p_{i2}^k, \cdots, p_{id}^k)^T$，速度为 $v_i^k = (v_{i1}^k, v_{i2}^k, \cdots, v_{id}^k)^T$，整个种群经过的最好位置为 $g_i^k = (g_{i1}^k, g_{i2}^k, \cdots, g_{id}^k)^T$。每个微粒根据如下公式更新速度和位置[61, 157]：

$$v_{ij}^{k+1} = wv_{ij}^k + c_1 R_1 (p_{ij}^k - x_{ij}^k) + c_2 R_2 (g_{ij}^k - x_{ij}^k) \tag{6.12}$$

$$v_{ij}^{k+1} = \begin{cases} v_{\max} & v_{ij}^{k+1} > v_{\max} \\ v_{ij}^{k+1} & v_{\min} \leqslant v_{ij}^{k+1} \leqslant v_{\max} \\ v_{\min} & v_{ij}^{k+1} \leqslant v_{\min} \end{cases} \tag{6.13}$$

$$x_{ij}^{k+1} = \begin{cases} 1 & R_3 < sig(v_{ij}^{k+1}) \\ 0 & \text{否则} \end{cases} \tag{6.14}$$

其中，w 为惯性权重，起着平衡全局和局部搜索的作用；c_1、c_2 为学习因子，c_1 反映了微粒飞行过程中所记忆的最好位置对微粒飞行速度的影响，被称为“认知系数”，c_2 则反映了整个微粒群所记忆的最好位置对飞行速度的影响[61]，又称为“社会学习系数”；R_1，R_2 和 R_3 为[0, 1]上的随机数，$sig(x)$是一个模糊函数，其定义为 $sig(x) = 1/(1+e^{-x})$。v_{ij}^{k+1}决定了微粒的位置分量取0或1的概率，即以 $sig(v_{ij}^{k+1})$ 的概率取1，以 $1 - sig(v_{ij}^{k+1})$ 的概率取0[61, 157]。

定义 6.9 适应值

将约束优化问题转化为无约束问题，一般可采用罚函数的方法，这里采用较简单的和形式的罚函数，令罚函数为

$$h = \sum_{i=1}^{m} \left| \min\{0,\ b_i - \sum_{j=1}^{n} a_{ij}x_j\} \right| \tag{6.15}$$

适应值为

$$f - Mh = \sum_{j=1}^{n} c_j x_j - M \sum_{i=1}^{m} \left| \min\{0,\ b_i - \sum_{j=1}^{n} a_{ij}x_j\} \right| \tag{6.16}$$

其中，M 为充分大的正数。

多维背包问题的元胞微粒群算法主要流程如下：

Step 1 确定种群规模 N，惯性权重因子 w，学习因子 c_1 和 c_2，$nc \leftarrow 0$（nc 为迭代步数或搜索次数）。

Step 2 初始化所有微粒的位置和速度：

每个微粒的位置由 $x_{ij}^0 = \begin{cases} 0 & R(0,\ 1) < 0.5 \\ 1 & \text{否则} \end{cases}$ 随机生成。

每个微粒的速度由 $v_{ij}^0 = v_{min} + R(0,\ 1)(v_{max} - v_{min})$ 随机生成。

其中，$i = 1,\ 2,\ \cdots,\ N$，$j = 1,\ 2,\ \cdots,\ n$，$R(0,\ 1)$ 表示[0, 1]上的随机数，v_{max}和 v_{min}表示最大速度和最小速度。

Step 3 计算每个微粒的适应值，记录当前的最好解。

Step 4 按元胞邻居的定义，在邻居范围内演化，记录最好解。

Step 5 比较更新每个微粒的最好位置和整个种群的最好位置。

Step 6 置 $nc \leftarrow nc+1$。

Step 7 若 $nc <$ 预定的迭代次数，更新每个微粒的位置和速度；转 Step 3。

Step 8 输出目前的最好解。

(2) 计算实验

选用"http://elib.zib.de/pub/mp-testdata/ip/sac94-suite/index.html"公布的典型的测试多维背包问题的算例，测试元胞微粒群算法的优化性能。有关算法参数设置的讨论可参考文献[148]。这里，给出相关的推荐值：群体规模 $N=100$；最大最小速度 v_{max} 和 v_{min} 分别设置为 4 和 -4；惯性权重 $w=1$；学习因子 $c_1=c_2=2$。

标准多维背包问题测试库中，共 55 个算例，元胞微粒群算法对每个实例均能达到最优解，表 6.7 是元胞微粒群算法在测试每一个多维背包算例时，记录达到最优解时的迭代次数和采用的邻居类型。

表 6.7 多维背包问题元胞微粒群算法的计算结果

测试数据	m	n	已知最优解	元胞微粒群算法	迭代次数	邻居类型
weing1.dat	2	28	141 278	141 278	18	扩展 Moore 邻居
weing2.dat	2	28	130 883	130 883	9	扩展 Moore 邻居
weing3.dat	2	28	95 677	95 677	13	扩展 Moore 邻居
weing4.dat	2	28	119 337	119 337	15	扩展 Moore 邻居
weing5.dat	2	28	98 796	98 796	18	扩展 Moore 邻居
weing6.dat	2	28	130 623	130 623	11	扩展 Moore 邻居
pb4.dat	2	29	95 168	95 168	14	扩展 Moore 邻居
weing7.dat	2	105	1 095 445	1 095 445	108	Moore 邻居
weing8.dat	2	105	624 319	624 319	101	Moore 邻居
hp1.dat	4	28	3 418	3 418	9	扩展 Moore 邻居
pb1.dat	4	27	3 090	3 090	17	扩展 Moore 邻居
pb2.dat	4	34	3 186	3 186	21	扩展 Moore 邻居
hp2.dat	4	35	3 186	3 186	28	扩展 Moore 邻居
weish01.dat	5	30	4 554	4 554	18	扩展 Moore 邻居

续表

测试数据	m	n	已知最优解	元胞微粒群算法	迭代次数	邻居类型
weish02.dat	5	30	4 536	4 536	27	扩展 Moore 邻居
weish03.dat	5	30	4 115	4 115	8	扩展 Moore 邻居
weish04.dat	5	30	4 561	4 561	13	扩展 Moore 邻居
weish05.dat	5	30	4 514	4 514	25	扩展 Moore 邻居
pet6.dat	5	39	10 618	10 618	32	扩展 Moore 邻居
weish06.dat	5	40	5 557	5 557	33	扩展 Moore 邻居
weish07.dat	5	40	5 567	5 567	17	扩展 Moore 邻居
weish08.dat	5	40	5 605	5 605	18	扩展 Moore 邻居
weish09.dat	5	40	5 246	5 246	12	扩展 Moore 邻居
weish10.dat	5	50	6 339	6 339	22	扩展 Moore 邻居
weish11.dat	5	50	5 643	5 643	19	扩展 Moore 邻居
weish12.dat	5	50	6 339	6 339	19	扩展 Moore 邻居
weish13.dat	5	50	6 159	6 159	33	扩展 Moore 邻居
pet7.dat	5	50	16 537	16 537	30	扩展 Moore 邻居
weish14.dat	5	60	6 954	6 954	37	扩展 Moore 邻居
weish15.dat	5	60	7 486	7 486	27	扩展 Moore 邻居
weish16.dat	5	60	7 289	7 289	37	扩展 Moore 邻居
weish17.dat	5	60	8 633	8 633	27	扩展 Moore 邻居
weish18.dat	5	70	9 580	9 580	61	Moore 邻居
weish19.dat	5	70	7 698	7 698	84	Moore 邻居
weish20.dat	5	70	9 450	9 450	103	Moore 邻居
weish21.dat	5	70	9 074	9 074	98	Moore 邻居
weish22.dat	5	80	8 947	8 947	87	Moore 邻居
weish23.dat	5	80	8 344	8 344	106	Moore 邻居

续表

测试数据	m	n	已知最优解	元胞微粒群算法	迭代次数	邻居类型
weish24.dat	5	80	10 220	10 220	107	Moore 邻居
weish25.dat	5	80	9 939	9 939	117	Moore 邻居
weish26.dat	5	90	9 584	9 584	55	Moore 邻居
weish27.dat	5	90	9 819	9 819	93	Moore 邻居
weish28.dat	5	90	9 492	9 492	79	Moore 邻居
weish29.dat	5	90	9 410	9 410	62	Moore 邻居
weish30.dat	5	90	11 191	11 191	77	Moore 邻居
pet2.dat	10	10	87 061	87 061	3	扩展 Moore 邻居
pet3.dat	10	15	4 015	4 015	11	扩展 Moore 邻居
pet4.dat	10	20	6 120	6 120	13	扩展 Moore 邻居
flei.dat	10	20	2 139	2 139	11	扩展 Moore 邻居
pb5.dat	10	20	2 139	2 139	13	扩展 Moore 邻居
pet5.dat	10	28	12 400	12 400	34	扩展 Moore 邻居
pb7.dat	30	37	1 035	1 035	32	Moore 邻居
pb6.dat	30	40	776	776	19	Moore 邻居
sent01.dat	30	60	7 772	7 772	49	Moore 邻居
sent02.dat	30	60	8 722	8 722	70	Moore 邻居

为进一步验证算法的性能，将算法与二进制微粒群算法[61]和文献[158]的禁忌搜索算法比较。为方便起见，元胞微粒群算法、二进制微粒群算法和禁忌搜索算法分别用 CPSO（Cellular Particle Swarm Optimization）、BPSO（Binary Particle Swarm Optimization）和 TS（Tabu Search）表示。元胞微粒群算法和二进制微粒群算法参数设置同前所述，元胞邻居均采用 Moore 邻居，禁忌搜索算法参数设置见文献[158]。每个算例独立运行 20 次，记录 20 次实验中获优次数、最优解、最劣解和平均解，实验结果如表 6.8 所示。图 6.2 和 6.3 是元胞微粒群算法和二进制微粒群算法部分算例寻优效果对比图。

表 6.8　元胞微粒群算法、二进制微粒群算法和禁忌搜索算法的性能比较

测试数据	m	n	最优解	算法	最大迭代次数	获优次数	最优解	最劣解	平均解
weing4.dat	2	28	119 337	BPSO	80	10/20	119 337	115 831	118 110
				TS	112	20/20	119 337	119 337	119 337
				CPSO	80	20/20	119 337	119 337	119 337
weing6.dat	2	28	130 623	BPSO	100	7/20	130 623	130 233	130 370
				TS	112	11/20	130 623	129 272	130 367.7
				CPSO	100	20/20	130 623	130 623	130 623
weing8.dat	2	105	624 319	BPSO	500	0/20	596 790	561 228	579 930
				TS	420	12/20	624 319	621 086	623 025.8
				CPSO	500	15/20	624 319	606 029	623 220
weish09.dat	5	40	5 246	BPSO	100	10/20	5 246	5 168	5 223.5
				TS	200	20/20	5 246	5 246	5 246
				CPSO	100	20/20	5 246	5 246	5 246
pet7.dat	5	50	16 537	BPSO	100	0/20	16 458	16 188	16 324
				TS	200	17/20	16 537	16 524	16 535
				CPSO	100	18/20	16 537	16 465	16 533
weish11.dat	5	50	5 643	BPSO	100	0/20	5 624	5 425	5 517.8
				TS	200	20/20	5 643	5 643	5 643
				CPSO	100	19/20	5 643	5 639	5 642.8
weish12.dat	5	50	6 339	BPSO	100	0/20	6 318	6 072	6 240.7
				TS	200	17/20	6 339	6 338	6 338.85
				CPSO	100	20/20	6 339	6 339	6 339
weish13.dat	5	50	6 159	BPSO	100	0/20	6 140	5 925	6 052.6
				TS	200	18/20	6 159	6 147	6 157.8
				CPSO	100	19/20	6 159	6 050	6 153.6

续表

测试数据	m	n	最优解	算法	最大迭代次数	获优次数	最优解	最劣解	平均解
weish14.dat	5	60	6 954	BPSO	150	0/20	6 923	6 732	6 819.9
				TS	240	18/20	6 954	6 921	6 950.8
				CPSO	150	18/20	6 954	6 923	6 950.9
weish15.dat	5	60	7 486	BPSO	100	0/20	7 449	7 128	7 318.9
				TS	240	20/20	7 486	7 486	7 486
				CPSO	100	20/20	7 486	7 486	7 486
weish16.dat	5	60	7 289	BPSO	100	0/20	7 269	7 021	7 124.5
				TS	240	20/20	7 289	7 289	7 289
				CPSO	100	19/20	7 289	7 288	7 288.9
weish17.dat	5	60	8 633	BPSO	150	0/20	8 618	8 519	8 577.6
				TS	240	19/20	8 633	8 624	8 632.55
				CPSO	150	19/20	8 633	8 624	8 632.55
weish18.dat	5	70	9 580	BPSO	200	0/20	9 549	9 364	9 454.9
				TS	280	19/20	9 580	9 573	9 579.65
				CPSO	200	17/20	9 580	9 565	9 577.8
weish19.dat	5	70	7 698	BPSO	200	0/20	7 683	7 340	7 546.4
				TS	280	20/20	7 698	7 698	7 698
				CPSO	200	17/20	7 698	7 683	7 695.8
weish20.dat	5	70	9 450	BPSO	200	0/20	9 408	9 154	9 308.9
				TS	280	17/20	9 450	9 430	9 447
				CPSO	200	18/20	9 450	9 433	9 448.3
weish25.dat	5	80	9 939	BPSO	200	0/20	9 801	9 553	9 678.5
				TS	320	17/20	9 939	9 936	9 938.55
				CPSO	200	16/20	9 939	9 923	9 937.1

续表

测试数据	m	n	最优解	算法	最大迭代次数	获优次数	最优解	最劣解	平均解
weish26.dat	5	90	9 584	BPSO	200	0/20	9 283	8 870	9 083.1
				TS	360	19/20	9 584	9 542	9 581.9
				CPSO	200	15/20	9 584	9 543	9 577.8
weish27.dat	5	90	9 819	BPSO	200	0/20	9 551	9 071	9 307.3
				TS	360	19/20	9 819	9 816	9 818.85
				CPSO	200	20/20	9 819	9 819	9 819
weish28.dat	5	90	9 492	BPSO	200	0/20	9 345	8 877	9 048.3
				TS	360	13/20	9 492	9 469	9 483.5
				CPSO	200	15/20	9 492	9 438	9 482.7
weish29.dat	5	90	9 410	BPSO	200	0/20	9 287	8 720	9 026
				TS	360	20/20	9 410	9 410	9 410
				CPSO	200	19/20	9 410	9 394	9 409.2
weish30.dat	5	90	11 191	BPSO	400	0/20	11 098	10 787	10 941
				TS	360	19/20	11 191	11 182	11 190.5
				CPSO	400	13/20	11 191	11 182	11 190
pet3.dat	10	15	4 015	BPSO	30	14/20	4 015	4 005	4 012
				TS	60	19/20	4 015	4 005	4 014
				CPSO	30	20/20	4 015	4 015	4 015
pet5.dat	10	28	12 400	BPSO	100	3/20	12 400	12 330	12 376
				TS	112	18/20	12 400	12 370	12 398
				CPSO	100	19/20	12 400	12 380	12 399
pb7.dat	30	37	1 035	BPSO	100	0/20	1 034	936	1 009.5
				TS	148	20/20	1 035	1 035	1 035
				CPSO	100	19/20	1 035	1 034	1 034.95

续表

测试数据	m	n	最优解	算法	最大迭代次数	获优次数	最优解	最劣解	平均解
pb6.dat	30	40	776	BPSO	100	7/20	776	679	724.5
				TS	160	20/20	776	776	776
				CPSO	100	20/20	776	776	776

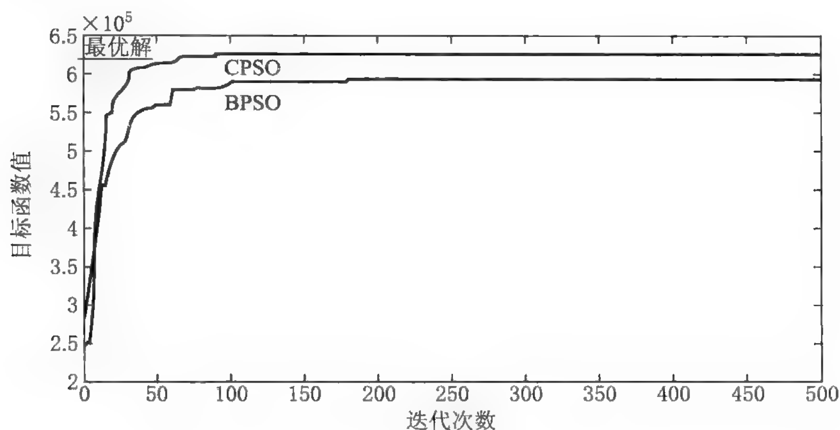


图 6.2　Weing8.dat(m=2, n=105)

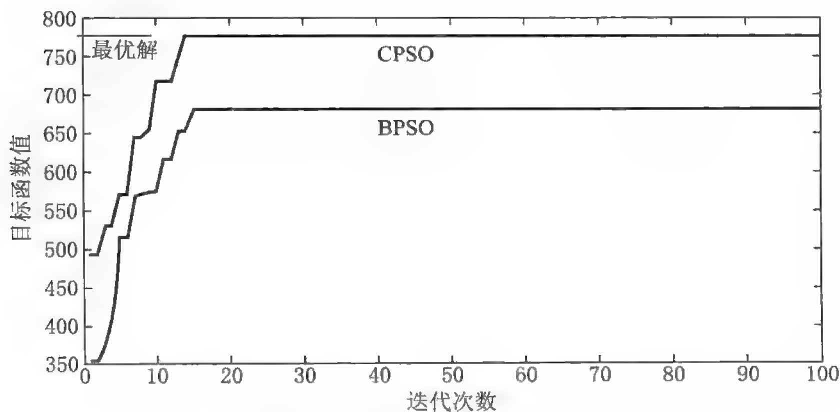


图 6.3　pb6.dat(m=30, n=40)

由表 6.8 的对比数据可以看出，不论获优次数、最优解的值还是最劣解的值、平均解的值，元胞微粒算法均远远优于二进制微粒群算法。从图 6.2 和 6.3 也可以看出，元胞微粒算法不仅在最终优化结果上，而且在收敛速度上相比于二进制微粒群算法都有显著的改善，可以更快地达到全局最优。

根据微粒群算法的原理[58-60]，微粒群算法的全局搜索能力强，但局部搜索能力弱。例如，二进制微粒群算法局部优化能力较弱，一旦陷入局部极值就无法摆脱，在增加迭代次数的情况下也无法避免。元胞微粒群算法是将元胞自动机原理和微粒群算法相融合的一种优化算法。在微粒群算法中，微粒之间进行优化信息的交流与共享，不断调整自己的位置，向最优解方向搜索[159]。而基于元胞自动机原理，重复简单的运算规则能描述复杂行为，反复进行局部的交互作用能实现全局优化。分布在元胞空间中的微粒在寻优的同时，按照元胞演化规则在邻居范围内作同步更新，避免算法早熟收敛，增强全局优化能力；每个微粒都具有感知能力[143]，通过优化信息共享机制调整搜索方向，逐渐逼近全局最优解。在元胞微粒算法中利用元胞及其邻居来保持种群的多样性，从而增强算法搜索的多样性；按演化规则寻优也有助于微粒发现较其目前个体最优更优的位置，从而提高其搜索速度，若发现较当前全局最优更优的位置，有利于提高整个种群的搜索速度。元胞微粒算法性能与禁忌搜索算法相当，部分算例的获优次数等指标优于禁忌搜索算法，除个别算例外，最大迭代次数要少于禁忌搜索算法。

6.2 元胞引力搜索算法

元胞蚁群算法和元胞微粒群算法的实验结果表明，元胞自动机在智能优化算法的应用是可行和有效的。采用现有的元胞自动机在智能优化算法应用的一些研究成果，以最小比率旅行商问题为例，给出一种元胞引力搜索算法。

6.2.1 算法设计

最小比率旅行商问题（Minimum Ratio Traveling Salesman Problem, MRTSP）是在经典旅行商问题（Traveling Salesman Problem，TSP）的基础上发展起来的[160]，旅行商经过城市 i 到 j 可以得到某种利益 p_{ij}（如商品销售的利润），现要求回路的总路程和总收益之比最小。MRTSP 的提法和经济学中的费用效益比类似，不仅考虑成本，而且考虑收益，比 TSP 中单纯的考虑总路程最短

更有意义。

本节假设考虑的都是完全图(否则,可通过求解任意两点之间的最短路化为等价的完全图)。令 $G=\{V, E\}$ 表示对称赋权完全图,$V=\{1, 2, \cdots, n\}$ 表示顶点集,$E=\{(i, j): i \neq j, i, j \in V\}$ 表示边集,距离矩阵 $D=(d_{ij})_{n\times n}$, $d_{ij}=d_{ji}$, $d_{ii}=+\infty$, $d_{ij}>0$, $(i, j \in V)$,收益矩阵 $P=(p_{ij})_{n\times n}$, $p_{ij}=p_{ji}$, $p_{ii}=0$, $p_{ij}>0(i, j \in V)$。$x_{ij}=1$ 表示边(i, j)在回路上,否则,$x_{ij}=0$。

MRTSP 的数学模型可以表示为:

$$\text{Min } Z=\frac{\sum_{i\neq j} d_{ij}x_{ij}}{\sum_{i\neq j} p_{ij}x_{ij}}$$

$$\text{s.t}\begin{cases}\sum_{i\neq j} x_{ij}=1 & j\in V\\ \sum_{j\neq i} x_{ij}=1 & i\in V\\ \sum_{i,\, j\in S} x_{ij}\leqslant |S|-1 & \forall S\subset V\\ x_{ij}\in\{0, 1\} & i, j\in V\end{cases} \tag{6.17}$$

其中,$|S|$表示集合 S 中含有图 G 中的顶点数,第一个和第二个约束要求每个顶点只有一条入边和进边,第三个约束要求没有任何子回路产生,因此,同时满足约束的解就形成了一条 Hamilton 回路。

元胞自动机最基本的组成部分为元胞、元胞空间、邻居和规则这四部分。有关邻居和规则的设置在元胞蚁群算法和元胞微粒群算法中已经取得成效,在元胞引力搜索算法中,也采用同样的方法,即定义 6.2 和 6.3 中给出的方法。元胞和元胞空间要根据具体问题进行设置,具体定义如下:

定义 6.10 给定城市元素的集合 $C=(c_1, c_2, \cdots, c_i, \cdots, c_n)$,其中 c_i 表示某个城市的编号。C 的任意排序组合的集合为元胞空间,可以表示为 $L=\{CellX=(c_1, \cdots, c_i, \cdots, c_j, \cdots, c_n) \mid c_i\in C, c_i\neq c_j, i, j=1, 2, \cdots, n\}$,其中每一个组合 $CellX$ 为元胞。

引力搜索算法最初是为求解连续优化问题而提出的,将算法扩展到求解离散优化问题时,实现算法搜索空间和问题解空间的对应是一个关键因素。本节采用的方法如下:

定义 6.11　编码设计

算法的搜索空间仍然是连续的，但将每个搜索变量的范围限制在[0，1]之间，并利用随机键的编码方法，将连续的个体位置转换为离散的城市访问顺序。

下面通过一个简单的例子来解释这个方法。例如，对于 5 个城市的 MRTSP，算法的解为(0.46，0.91，0.33，0.75，0.51)，按照随机键的方法，这 5 个城市的访问顺序为 4→1→5→2→3。

综上所述，元胞引力搜索算法的流程如下：

步骤 1: 参数初始化。

步骤 2: 对个体位置进行编码。

步骤 3: 计算个体质量和所受的引力。

步骤 4: 更新加速度，速度和位置。

步骤 5: 根据元胞邻居的定义，在邻居范围内进行演化，并保存当前最优解。

步骤 6: 若未达到最大迭代次数，则返回步骤 2；否则，算法停止，输出结果。

6.2.2　实例计算

为验证算法的性能，采用文献[55]和[161]中的算例进行测试，并与基本引力搜索算法的计算结果进行比较。两种算法群体规模设为 50，最大迭代次数设为 500，其他参数设置采用文献[63]的推荐值。

例 6.7　设给定对称赋权完全图 $G=\{V, E\}$，距离矩阵 D 和收益矩阵 P 定义如下：

$$D=\begin{bmatrix} \infty & 15 & 62 & 27 & 19 \\ 15 & \infty & 75 & 69 & 71 \\ 62 & 75 & \infty & 47 & 49 \\ 27 & 69 & 47 & \infty & 71 \\ 19 & 71 & 49 & 71 & \infty \end{bmatrix}$$

$$P=\begin{bmatrix} 0 & 32 & 20 & 72 & 62 \\ 32 & 0 & 28 & 45 & 30 \\ 20 & 28 & 0 & 77 & 18 \\ 72 & 45 & 77 & 0 & 10 \\ 62 & 30 & 18 & 10 & 0 \end{bmatrix}$$

例 6.8　设给定对称赋权完全图 $G=\{V, E\}$，距离矩阵 D 和收益矩阵 P 定义如下：

$$D=\begin{bmatrix} \infty & 9 & 48 & 17 & 68 & 52 & 57 & 95 \\ 9 & \infty & 16 & 83 & 94 & 28 & 100 & 9 \\ 48 & 16 & \infty & 14 & 59 & 61 & 62 & 47 \\ 17 & 83 & 14 & \infty & 69 & 7 & 24 & 64 \\ 68 & 94 & 59 & 69 & \infty & 43 & 84 & 31 \\ 52 & 28 & 61 & 7 & 100 & \infty & 43 & 35 \\ 57 & 100 & 62 & 24 & 84 & 43 & \infty & 6 \\ 95 & 9 & 47 & 64 & 31 & 35 & 6 & \infty \end{bmatrix}$$

$$P=\begin{bmatrix} 0 & 9 & 46 & 62 & 45 & 20 & 75 & 24 \\ 9 & 0 & 8 & 57 & 78 & 18 & 9 & 60 \\ 46 & 8 & 0 & 21 & 4 & 41 & 48 & 33 \\ 62 & 57 & 21 & 0 & 39 & 94 & 29 & 14 \\ 45 & 78 & 4 & 39 & 0 & 36 & 86 & 2 \\ 20 & 18 & 41 & 94 & 36 & 0 & 41 & 23 \\ 75 & 9 & 48 & 29 & 86 & 41 & 0 & 3 \\ 24 & 60 & 33 & 14 & 2 & 23 & 3 & 0 \end{bmatrix}$$

例 6.9　设给定对称赋权完全图 $G=\{V, E\}$，距离矩阵 D 和收益矩阵 P 定义如下：

$$D=\begin{bmatrix} \infty & 1 & 862 & 273 & 319 & 373 & 83 & 71 & 60 & 918 \\ 1 & \infty & 775 & 698 & 718 & 163 & 467 & 826 & 482 & 875 \\ 862 & 775 & \infty & 773 & 493 & 828 & 142 & 501 & 593 & 775 \\ 273 & 698 & 773 & \infty & 771 & 558 & 682 & 956 & 999 & 677 \\ 319 & 718 & 493 & 771 & \infty & 86 & 496 & 510 & 954 & 339 \\ 373 & 163 & 828 & 558 & 86 & \infty & 706 & 775 & 198 & 914 \\ 83 & 467 & 142 & 682 & 496 & 706 & \infty & 590 & 690 & 94 \\ 71 & 826 & 501 & 956 & 510 & 775 & 590 & \infty & 281 & 162 \\ 60 & 482 & 593 & 999 & 594 & 198 & 690 & 281 & \infty & 544 \\ 918 & 875 & 775 & 677 & 339 & 914 & 94 & 162 & 544 & \infty \end{bmatrix}$$

$$
P=\begin{bmatrix}
0 & 32 & 203 & 672 & 162 & 426 & 475 & 841 & 294 & 368 \\
32 & 0 & 328 & 845 & 307 & 330 & 247 & 280 & 150 & 288 \\
203 & 328 & 0 & 917 & 888 & 21 & 144 & 22 & 10 & 651 \\
672 & 845 & 917 & 0 & 709 & 207 & 593 & 645 & 244 & 296 \\
162 & 307 & 888 & 709 & 0 & 773 & 882 & 574 & 685 & 8 \\
426 & 330 & 21 & 207 & 773 & 0 & 989 & 750 & 144 & 785 \\
475 & 247 & 144 & 593 & 882 & 989 & 0 & 865 & 288 & 629 \\
841 & 280 & 22 & 645 & 574 & 750 & 865 & 0 & 878 & 274 \\
294 & 150 & 10 & 244 & 685 & 144 & 288 & 878 & 0 & 961 \\
368 & 288 & 651 & 296 & 8 & 785 & 629 & 274 & 961 & 0
\end{bmatrix}
$$

三个算例目前已知的最优值分别为 0.850 43，0.639 10 和 0.408 71，为方便比较，计算结果保留相同的位数。对这 3 个算例分别独立运行 10 次，记录每次的计算结果和总的获优次数，具体结果如表 6.9 所示。元胞引力搜索算法和引力搜索算法分别用 CGSA（Cellular Gravitational Search Algorithm）和 GSA（Gravitational Search Algorithm）表示。

表 6.9　MRTSP 的实验结果

数　据	最优值	算法	测　试　结　果					成功次数
算例 1	0.850 43	GSA	0.850 43	0.850 43	0.850 43	0.850 43	0.850 43	10/10
			0.850 43	0.850 43	0.850 43	0.850 43	0.850 43	
		CGSA	0.850 43	0.850 43	0.850 43	0.850 43	0.850 43	10/10
			0.850 43	0.850 43	0.850 43	0.850 43	0.850 43	
算例 2	0.639 10	GSA	0.639 10	0.658 87	0.639 10	0.639 10	0.658 87	8/10
			0.639 10	0.639 10	0.639 10	0.639 10	0.639 10	
		CGSA	0.639 10	0.639 10	0.639 10	0.639 10	0.639 10	10/10
			0.639 10	0.639 10	0.639 10	0.639 10	0.639 10	
算例 3	0.408 71	GSA	0.483 29	0.408 71	0.483 29	0.408 71	0.408 71	7/10
			0.408 71	0.408 71	0.482 79	0.408 71	0.408 71	
		CGSA	0.408 71	0.408 71	0.408 71	0.408 71	0.408 71	10/10
			0.408 71	0.408 71	0.408 71	0.408 71	0.408 71	

MRTSP 的目标函数是非线性的，比求解目标函数是线性的 TSP 更为困难。从这几个算例的计算结果可以看出，元胞引力搜索算法的求解效果远好于引力搜索算法，每次都可以获得最优解，算法具有较好的优化精度和稳定性。元胞自动机和引力搜索算法的结合，也同样是可行有效的。在元胞引力搜索算法中，基于万有引力定律的优化机制引导群体进行全局搜索，而基于元胞邻居内的搜索有助于提高算法的局部开发能力，能够有效解决基本引力搜索算法局部优化能力较弱的问题。

第七章 结论与展望

7.1 结论

引力搜索算法是一种新兴的智能优化算法，其独特的优化机制为智能优化算法的发展注入了新的活力。但算法自提出至今，只有不到5年的发展时间，其研究还处于起步阶段。对单目标连续优化的研究还不多，在多目标连续优化方面的研究则更少，还没有对算法的理论研究。本书从理论和实验两方面对算法进行研究，主要工作如下：

(1) 综述了引力搜索算法的起源、发展以及应用情况，总结了目前引力搜索算法的研究中在理论基础和算法改进及应用方面的一些不足，指出了引力搜索算法一些可行的研究方向。

(2) 对用于单目标连续优化的基本引力搜索算法展开了研究。采用随机泛函分析中的随机压缩映射定理，对基本引力搜索算法的收敛性进行了分析，证明了算法在无限次迭代后能够收敛到全局最优解。但现有的研究成果表明，在有限的迭代次数内，基本引力搜索算法的全局探索能力较强而局部开发能力较弱，会存在求解精度不高和收敛速度慢等问题。在实际应用中通常要求在有限的迭代次数内，算法具有较高的优化精度和较快的收敛速度。因此，需要对基本引力搜索算法进行改进，以提高算法的优化性能。受微粒群优化算法的启发，在基本引力搜索算法位置更新方程中，引入了一个取值随迭代次数线性递减的系数。采用多个标准测试函数验证了新算法的可行性和有效性。

(3) 结合多目标连续优化问题的特点和引力搜索算法的特征，给出了一种求解多目标优化问题的引力搜索算法。该算法主要操作有非支配排序、拥挤距离计算、新个体与被删除个体的比较和引力搜索算法中的基本操作。从理论上证明了所设计出的算法具有全局收敛性，并对此进行数值实验验证。所提算法在求解多目标连续优化问题时表现出良好的优化效果。

(4) 采用基于引力搜索算法的方法求解了一些实际问题，并取得了不错的结果。这些问题包括投资者偏好条件下概率准则投资组合问题、非线性极大极小问题、系统可靠性优化问题和多目标排水控制问题。虽然基本引力搜索算法具有全局优化的优点，但是局部搜索的能力较弱。在基本引力搜索算法中结合局部优化方法是改进算法求解性能的一种有效途径。在处理这些实际问题时，采用了混沌优化方法和序列二次规划等方法。计算结果表明，基本引力搜索算法结合局部搜索算法后，求解质量都有显著提高。

(5) 在分析元胞蚁群算法和元胞微粒群算法的基础上，以最小比率旅行商问题为例，提出了一种元胞引力搜索算法的计算方法。将引力搜索算法从连续优化领域中的应用延伸到离散优化领域，为算法的进一步应用研究提供了一些思路。

本书的研究工作充实了引力搜索算法的理论基础，进一步加强了引力搜索算法求解连续优化问题的能力，拓展了引力搜索算法在处理离散空间问题中的求解能力，对引力搜索算法的发展具有一定的理论和现实意义。

7.2 展望

虽然在研究过程中，取得了一些研究成果。但是，作为一种新型智能优化算法，引力搜索算法还有很大的发展空间，今后的研究工作将在以下两个方面展开：

(1) 算法的完善和改进

引力搜索算法是物理学定律和优化思想相融合而得到的一种新型算法，属于交叉性的研究方向。要充分吸收这些方面的研究成果，用来更加有效地设计算法的搜索机制，进一步提高算法的优化性能。另外，还可以利用现有的优化算法，包括经典优化算法和智能优化算法，结合引力搜索算法提出改进的混合优化方法。此外，目前是根据实验进行算法参数的设置，如何从理论上给出算法参数设置的指导，也是需要研究的问题。

(2) 拓宽算法的应用领域

目前，引力搜索算法已经用于求解一些优化问题，并取得较为满意的结果。今后将更加注重引力搜索算法在一些未涉及的实际问题中的应用，如应急管理、智能电网和城市交通等，为这些问题的求解提供可行有效的方法。

参考文献

[1] 陈宝林.最优化理论与算法[M].北京:清华大学出版社,2003:1—8.

[2] 王宜举,修乃华.非线性最优化理论与方法[M].北京:科学出版社,2012:1—246.

[3] 袁亚湘.非线性优化计算方法[M].北京:科学出版社,2008:1—260.

[4] 李董辉,童小娇,万中.数值最优化算法与理论[M].北京:科学出版社,2010:1—289.

[5] 谢政,李建平,陈挚.非线性最优化理论与方法[M].北京:高等教育出版社,2010:1—336.

[6] 汪定伟,王俊伟,王洪峰,等.智能优化方法[M].北京:高等教育出版社,2007:1—19.

[7] 徐宗本,张讲社,郑亚林.计算智能中的仿生学:理论与算法[M].北京:科学出版社,2003:1—10.

[8] Kennedy J, Eberhart R C, Shi Y. Swarm intelligence[M]. San Francisco: Morgan Kaufmann Publishers, 2001:1—512.

[9] Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: from natural to artificial systems[M]. Oxford: Oxford University Press, 1999:1—320.

[10] Gazi V, Passino K M. Stability analysis of social foraging swarms[J]. IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics, 2004, 34(1):539—557.

[11] Warburton K, Lazarus J. Tendency-distance models of social cohesion in animal groups[J]. Journal of Theoretical Biology, 1991, 150(4):473—488.

[12] Breder C M. Equations descriptive of fish schools and other animal aggregations[J]. Ecology, 1954, 35(3):361—370.

[13] Engelbrecht A P. Computational intelligence: an introduction[M]. New Jersey: John Wiley & Sons Publishers, 2007:1—628.

[14] Floreano D, Mattiussi C. Bio-inspired artificial intelligence: theories, methods, and technologies[M]. Massachusetts: The MIT Press, 2008:1—659.

[15] Blum C, Merkle D. Swarm intelligence: introduction and applications[M]. Berlin: Springer-Verlag, 2008:1—286.

[16] Mogilner A, Edelstein-Keshet L. A non-local model for a swarm[J]. Journal of Mathematical Biology, 1999, 38(6):534—570.

[17] Gazi V, Passino K M. Stability analysis of swarms[J]. IEEE Transactions on Automatic Control, 2003, 48(4):692—697.

[18] Yang X S. Nature-inspired metaheuristic algorithms[M]. Beckington: Luniver Press, 2010:1—128.

[19] Engelbrecht A P. Fundamentals of computational swarm intelligence[M]. New Jersey: John Wiley & Sons Publishers, 2005:1—672.

[20] Okubo A. Dynamical aspects of animal grouping: swarms, schools, flocks, and herds[J]. Advances in Biophysics, 1986, (22):1—94.

[21] Giulietti F, Pollini L, Innocenti M. Autonomous formation flight[J]. IEEE Control Systems Magazine, 2000, 20(6):34—44.

[22] Czirok A, Vicsek T. Collective behavior of interacting self-propelled particles[J]. Physica A, 2000, 281(1—4):17—29.

[23] Aldana M, Huepe C. Phase transitions in self driven many-particle systems and related non-equilibrium models: a network approach[J]. Journal of Statistical Physics, 2003, 112(1—2):135—153.

[24] Couzin I D, Krause J, James R, et al. Collective memory and spatial sorting in animal groups[J]. Journal of Theoretical Biology, 2002, 218(1):1—11.

[25] Topaz C M, Bertozzi A L. Swarming patterns in a two-dimensional kinematics model for biological groups[J]. SIAM Journal on Applied Mathematics, 2005, 65(1):152—174.

[26] Holland J H. Adaptation in natural and artificial systems[M]. Massachusetts: The MIT Press, 1992:1—228.

[27] Kirkpatrick S, Gelatt C D, Vecchi M P.Optimization by simulated annealing[J]. Science, 1983, 220(4598):671—680.
[28] 吴微.神经网络计算[M].北京:高等教育出版社,2004:1—79.
[29] Dorigo M, Maniezzo V, Colorni A.Ant system: optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 1996, 26(1):29—41.
[30] Kennedy J, Eberhart R C.Particle swarm optimization[C]. Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, IEEE Press, 1995:1942—1948.
[31] King R L, Russ S H, Lambert A B, et al. Anartificial immune system model for intelligent agents[J]. Future Generation Computer Systems, 2001, 17(4): 35—343.
[32] Hofmeyr S A, Forrest S.Architecture for an artificial immune system[J]. Evolutionary computation, 2000, 8(4):443—473.
[33] Jung S H.Queen-bee evolution for genetic algorithms[J]. Electronics Letters, 2003, 9(6):575—576.
[34] Tereshko V, Loengarov A.Collective decision making in honey-bee foraging dynamics[J]. Computing and Information Systems, 2005, 9(3):1—7.
[35] Simon D.Biogeography-Based Optimization[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(6):702—713.
[36] Simon D.A probabilistic analysis of a simplified biogeography-based optimization algorithm[J]. Evolutionary Computation, 2011, 19(2):167—188.
[37] 马良.基础运筹学教程[M].北京:高等教育出版社,2006:263—265.
[38] Lawrence J.Introduction to Neural Networks[M]. California: California Scientific Software Press, 1994:1—120.
[39] 史忠植.神经网络[M].北京高等教育出版社,2009:1 —80.
[40] 蒋宗礼.人工神经网络导论[M].高等教育出版社,2008:1—50.
[41] Haykin S.Neural Networks: A Comprehensive Foundation[M]. New Jersey: Prentice Hall, 1999:1—55.
[42] 王凌.智能优化算法及其应用[M].北京:清华大学出版社,2001:83 —90.

[43] 康立山,谢云,尤矢勇,等.非数值并行算法(第一册)——模拟退火算法[M].北京:科学出版社,1997:22—38.

[44] 张军,詹志辉.计算智能[M].北京:清华大学出版社,2009:1—216.

[45] Granville V, Krivanek M, Rasson J P.Simulated annealing: A proof of convergence[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994, 16(6):652—656.

[46] Cerny V.Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm[J]. Journal of Optimization Theory and Applications, 1985, 45(1):41—51.

[47] 康立山,陈毓屏.解货郎担问题的异步并行模拟退火算法[J].自然科学进展,1991, 1(3):246—252.

[48] Vose M D. The Simple genetic algorithm: foundations and theory[M]. Massachusetts: the MIT Press, 1999:1—50.

[49] 刘勇,康立山,陈毓屏.非数值并行算法(第二册)——遗传算法[M].北京:科学出版社,1998:1—33.

[50] Mitchell M.An introduction to genetic algorithms[M]. Massachusetts: the MIT Press, 1996:1—56.

[51] 李郝林.遗传算法中的逐级进化策略研究[J].上海理工大学学报,2004, 26(2):138—140.

[52] Goldberg D E.Genetic algorithms in search, optimization and machine learning[M]. Reading: Addison Wesley, 1989:1—10.

[53] Dorigo M, Gambardella L M.Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1):53—56.

[54] Colorni A, Dorigo M, Maniezzo V.Distributed optimization by ant colonies[C]. Proceedings of the First European Conference on Artificial Life, Paris, Elsevier, 1992:134—142.

[55] 马良,朱刚,宁爱兵.蚁群优化算法[M].北京:科学出版社,2008:1—252.

[56] 马良,项培军.蚂蚁算法在组合优化中的应用[J].管理科学学报,2004, 4(2):32—37.

[57] 段海滨,张祥银,徐春芳.仿生智能计算[M].北京:科学出版社,2011:1 286.

[58] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]. Proceedings Sixth Symposium on Micro Machine and Human Science. Nagoya, IEEE Press, 1995:39—43.

[59] Shi Y, Eberhart R. A modified particle swarm optimizer[C]. Proceedings of the 1998 IEEE Conference on Evolutionary Computation, Anchorage, IEEE Press, 1998:69—73.

[60] Shi Y, Eberhart R C. Empirical study of particle swarm optimization[C]. Proceedings of the 1999 Congress on Evolutionary Computation Piscataway, IEEE Press, 1999:1945—1950.

[61] 曾建潮,介婧,崔志华.微粒群算法[M].北京:科学出版社,2004:1 -8.

[62] 韩毅,唐加福,牟立峰,等.粒子群算法求解无能力约束生产批量计划问题[J].管理科学学报,2008, 11(5):33—40.

[63] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm[J]. Information Science, 2009, 179(13):2232—2248.

[64] Rashedi E, Nezamabadi-pour H, Saryazdi S. BGSA: binary gravitational search algorithm[J]. Natural Computing, 2010, 9(3):727—745.

[65] Schutz B. Gravity from the ground up[M]. Cambridge: Cambridge University Press, 2003:1—48.

[66] Holliday D, Resnick R, Walker J. Fundamentals of physics[M]. New Jersey: John Wiley & Sons Publishers, 1993:20—89.

[67] 方福康,狄增如.钱学森与系统科学基础理论的发展[J].上海理工大学学报,2011, 33(6):565—568.

[68] Holland J H. Hidden order: how adaptation builds complexity[M]. Massachusetts: Addison-Wesley, 1995:1—55.

[69] 许国志.系统科学[M].上海:上海科技教育出版社,2000:1 -413.

[70] Sarafrazi S, Nezamabadi-pour H, Saryazdi S. Disruption: a new operator in gravitational search algorithm[J]. Scientia Iranica, 2011, 18(3):539—548.

[71] 徐遥,王士同.引力搜索算法的改进[J].计算机工程与应用,2011, 47(35):188—192.

[72] 徐遥,安亚静,王上同.基于三角范数的引力搜索算法分析[J].计算机科学,2011,38(11):225—230.

[73] Mirjalili S, Hashim A Z M. A new hybrid PSOGSA algorithm for function optimization[C]. International Conference on Computer and Information Application, Tianjin, IEEE Press, 2010:374—377.

[74] Li X G, Yin M H, Ma Z Q. Hybrid differential evolution and gravitation search algorithm for unconstrained optimization[J]. International Journal of the Physical Sciences, 2011, 6(25):5961—5981.

[75] Hassanzadeh H R, Rouhani M. A multi-objective gravitational search algorithm [C]. 2010 Second International Conference on Computational Intelligence, Communication Systems and Networks, Liverpool, IEEE Press, 2010:7—12.

[76] Nobahari H, Nikusokhan M, Siarry P. Non-dominated sorting gravitational search algorithm[C]. International conference on swarm intelligence, Cergy, 2011:1—10.

[77] Hatamlou A, Abdullah S, Nezamabadi-pour H. Application of gravitational search algorithm on data clustering[C]. The Sixth International Conference of Rough Set and Knowledge Technology, Banff, Springer-Verlag, 2011:337—346.

[78] Yin M H, Hu Y M, Yang F Q, et al. A novel hybrid K-harmonic means and gravitational search algorithm approach for clustering[J]. Expert Systems with Applications, 2011, 38(8):9319—9324.

[79] Hatamlou A, Abdullah S, Othman Z. Gravitational search algorithm with heuristic search for clustering problems[C]. The Third Conference on Data Mining and Optimization, Selangor, IEEE Press, 2011:190—193.

[80] Duman S, Maden D, Guvenc U. Determination of the PID controller parameters for speed and position control of DC motor using gravitational search algorithm [C]. The seventh International Conference on Electrical and Electronics Engineering, Bursa, IEEE Press, 2011:225—229.

[81] Affijulla S, Chauhan S. A new intelligence solution for power system economic load dispatch[C]. The Tenth International Conference on Environment and Elec-

trical Engineering, Rome, IEEE Press, 2011:1—5.
[82] Shaw B, Mukherjee V, Ghoshal S P. A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems[J]. Electrical Power and Energy Systems, 2012, 35(1):21—33.
[83] Papa J P, Pagnin A, Schellini S A, et al. Feature selection through gravitational search algorithm[C]. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Prague, IEEE Press, 2011: 2052—2055.
[84] Xiao J H, Cheng Z.DNA sequences optimization based on gravitational search algorithm for reliable DNA computing[C]. The Sixth International Conference on Bio-Inspired Computing: Theories and Applications, Penang, IEEE Press, 2011:103—107.
[85] 谷文祥,李向涛,朱磊,等.求解流水线调度问题的引力搜索算法[J].智能系统学报,2010, 5(5):411—418.
[86] Zhao W G.Adaptive Image enhancement based on gravitational search algorithm [J]. Procedia Engineering, 2011, 15:3288—3292.
[87] Zibanezhad B, Zamanifara K, Sadjady R S, et al. Applying gravitational search algorithm in the QoS-based Web service selection problem[J]. Journal of Zhejiang University-Science C(Computers & Electronics), 2011, 12(9):730—742.
[88] Li C S, Zhou J Z.Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm[J]. Energy Conversion and Management 2011, 52(1):374—381.
[89] Chatterjee A, Mahanti G K.Design of fully digital controlled reconfigurable dual-beam concentric ring array antenna using GSA[J]. Progress In Electromagnetics Research C, 2011, 18:59—72.
[90] Khademolghorani F, Baraani A, Zamanifar K.Efficient mining of association rules based on gravitational search[J]. IJCSI International Journal of Computer Science Issues, 2011, 8(4):51—58.
[91] Behrang M A, Assareh E, Ghalambaz M, et al. Forecasting future oil demand in Iran using GSA[J]. Energy, 2011, 36(9):5649—5654.

[92] Rashedi E, Nezamabadi-pour H, Saryazdi S.Filter modeling using gravitational search algorithm[J]. Engineering Applications of Artificial Intelligence, 2011, 24(1):117—122.

[93] 卢同善.随机泛函分析及应用[M].青岛:青岛海洋大学出版社,1990:1—194.

[94] 王梓坤.随机泛函分析引论[J].数学进展,1962, 5(1):45—71.

[95] 金林鹏,李均利,魏平,等.用于函数优化的最大引力优化算法[J].模式识别与人工智能,2010, 23(5):653—662.

[96] 许天周.应用泛函分析[M].北京:科学出版社,2006:1—300.

[97] Clerc M, Kennedy J.The particle swarm-explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1):58—73.

[98] Yao X, Liu Y, Lin G.Evolutionary programming made faster[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(2):82—102.

[99] Chiong R, Weise T, Michalewicz Z.Variants of Evolutionary algorithms for real-world applications[M]. Berlin: Springer-Verlag, 2012:15—16.

[100] Back T.Evolutionary Algorithms in Theory and Practice[M]. Oxford: Oxford University Press, 1996:1—228.

[101] 焦李成,公茂果,尚荣华,等.多目标免疫优化算法、理论和应用[M].北京:科学出版社,2010:1—302.

[102] 崔逊学.多目标进化算法及其应用[M].北京:国防工业出版社,2008:1—305.

[103] 郑金华.多目标进化算法及其应用[M].北京:科学出版社,2007:1—251.

[104] Gonzalez-alvarez D L, Vega-Rodriguez M A, Gomez-Pulido J A, et al. Applying a multiobjective gravitational search algorithm(MO-GSA) to discover motifs[C]. 11th International Work-Conference on Artificial Neural Networks, Torremolinos-Malaga, Springer-Verlag, 2011:372—379.

[105] Rubio-largo A, Vega-Rodriguez M A, Gomez-Pulido J A, et al. A Multiobjective gravitational search algorithm applied to the static routing and wavelength assignment problem[C]. Proceedings of the 2011 international conference on Applications of evolutionary computation, Torino, Springer-Verlag, 2011:41—50.

[106] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multi-objective genetic algorithm: NSGA-II[J]. IEEE Transaction on Evolutionary Computation, 2002, 6(2):181—197.

[107] Abraham A, Ruiz-del-Solar J, Koppen M. Soft computing systems: design, management and applications[M]. Amsterdam: IOS Press, 2002:23—78.

[108] Aittokoski T, Miettinen K. Efficient evolutionary method to approximate the Pareto optimal set in multiobjective optimization[C]. International Conference on Engineering Optimization, Rio de Janeiro, 2008:1—10.

[109] Veldhuizen D A V. Multiobjective evolutionary algorithms classifications, analyses, and new innovations [D]. Ohio, USA: Air Force Institute of Technology, 1999.

[110] Coello C A C, Veldhuizen D A V, Lamont G B. Evolutionary Algorithms for Solving Multi-Objective Problems[M]. New York: Kluwer Academic Publishers, 2002:50—121.

[111] Veldhuizen D A V, Lamont G B. Evolutionary computation and convergence to a Pareto front[C]. Proceedings of the Third Annual Conference, San Francisco, 1998:22—25.

[112] Markowitz H M. Portfolio selection[J]. Journal of Finance, 1952, 7(1):77—91.

[113] 刘津振.组合证券投资策略[J].预测,1995, 14(1):60—61.

[114] 余峰,田益祥,李成刚,等.基于自由现金流量的证券投资策略及实证[J].预测,2011, 30(2):57—61.

[115] 梁建峰,唐万生.有交易费用的组合证券投资的概率准则模型[J].系统工程,2001, 19(2):5—10.

[116] Tang W S, Han Q H, Li G Q. Cash management decision with probability criterion[C]. 2001 IEEE International Conference on Systems, Man, and Cybernetics, Tucson, IEEE Press, 2001:2670—2673.

[117] 唐万生,梁建峰,韩其恒.组合证券投资的概率准则模型[J].系统工程学报,2004, 19(2):193—197.

[118] 王燕青,唐万生,韩其恒.基于遗传算法的概率准则组合证券模拟求解.管理科学学报[J], 2002, 5(6):29—33.

[119] 张莉,唐万生,宋军.概率准则下组合投资的整数规划模型[J].天津大学学报,2003, 5(2):126—128.

[120] 傅荣林.组合证券投资的概率准则模型探讨[J].运筹与管理,2002, 11(4):97—105.

[121] 丁元耀.概率风险准则下组合投资决策[J].统计与决策,2003,(3):22—23.

[122] 徐金虹,徐维军,李引生.概率原则下基于投资者偏好的组合投资选择[J].系统工程学报,2005, 20(5):544—548.

[123] 黄震宇,沈祖和.解一类非线性极大极小问题的熵函数方法[J].科学通报,1999, 41(17):1550—1554.

[124] 李坤杰,陶跃钢,刘国平.非线性极大极小系统全局优化算法的分析[J].数学的实践与认识,2008, 38(20):127—133.

[125] 李兵,蒋慰.混沌优化方法及其应用[J].控制理论与应用,1997, 14(4):613—615.

[126] Riccardo C, Luigif F, Stefano F, et al. Chaotic sequences to improve the performance of evolutionary algorithms[J]. IEEE Transactions on Evolutionary Computation, 2003, 7(3):289—304.

[127] Bilal A, Erhan, Bedri O.Chaos embedded particle swarm optimization algorithm[J]. Chaos, Solutions and Fractals, 2009, 40(4):1715—1734.

[128] 李兴斯.解非线性极大极小问题的凝聚函数法[J].计算结构力学及其应用,1991, 8(1):85—92.

[129] 纪震,廖惠连,吴青华.粒子群算法及应用[M].北京:科学出版社,2009:108—110.

[130] 文新辉,陈开周.解无约束极大极小问题的非对称神经网络算法[J].电子学报,1995, 23(12):111—114.

[131] 宋何维.系统可靠性设计与分析[M].西安:西北工业大学出版社,2008:1—56.

[132] 邢文训,谢金星.现代优化计算方法[M].北京:清华大学出版社,2005:1—41.

[133] 肖人彬,陶振武.群集智能研究进展[J].管理科学学报,2007, 10(3):80—96.

[134] Ravi V, Murty B S N, Reddy PJ.Nonequilibrium simulated annealing algorithm applied to reliability optimization of complex systems[J]. IEEE Transactions on

Reliability, 1997, 46(2):233—239.

[135] 许传玉,朱若男,梁颖虹,等.遗传算法在复杂系统可靠性优化中的应用[J].哈尔滨理工大学学报,2000, 5(3):90—93.

[136] Shelokar P S, Jayaraman V K, Kulkarni B D. Ant algorithm for single and multiobjective reliability optimization problems[J]. Quality and Engineering International, 2002, 18(6):497—514.

[137] Fletcher R. Practical Methods of Optimization[M]. New York: Wiley, 1987: 23—89.

[138] Boggs P T, Tolle J W. Sequential quadratic programming[J]. Acta Numerica, 1995, (4):1—51.

[139] Michael B B. Sequential quadratic programming[J]. Springer Optimization and Its Application, 2008, 19:1—14.

[140] 刘士新,宋健海,唐加福.蚁群最优化——模型、算法及应用综述[J].系统工程学报,2004, 19(5):496—502.

[141] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm [J]. Journal of Global Optimization, 2007, 39(3):459—471.

[142] Karaboga D, Basturk B. On the performance of artificial bee colony(ABC) algorithm[J]. Applied Soft Computing, 2008, 8(1):687—697.

[143] 殷国玺,张展羽,张国华,等.基于粒子群优化算法的农田多目标控制排水模型[J].农业工程学报,2009, 25(3):6—9.

[144] Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3):256—279.

[145] Von Neumann J, Burks A W. Theory of self reproducing automata [M]. Urbana: University of Illinois Press, 1966:32—45.

[146] 张永安,白学志.复杂系统研究的重要工具——细胞自动机及其应用[J].自然杂志,1997, 192—195, 196.

[147] 刘勇,马良.非线性 0—1 规划的元胞蚁群算法[J].系统管理学报,2010, 19(3):351—355.

[148] 刘勇，马良.元胞微粒群算法及其在多维背包问题中的应用[J].管理科学学报，2011，14(1)：86—96.

[149] 谢惠明.复杂性与动力系统[M].上海：上海教育科技出版社，1994：13—19.

[150] 黎夏，叶嘉安，刘小平，等.地理模拟系统：元胞自动机与多智能体[M].北京：科学出版社，2007：56—122.

[151] 周光明，王奇生，邓康.带线性约束 0—1 二次规划罚函数的改进[J].南华大学学报(理工版)，2004，18(1)：67—69.

[152] 刘凌，张圣贵.0—1 多项式规划的一种代数算法[J].福建师范大学学报(自然科学版)，2001，17(4)：22—25.

[153] Zhou X W, Wang Y Y, Tian X X, et al. A Tabu Search Algorithm for Quadratic 0—1 Programming Problem [J]. Chinese Quarterly Journal of Mathematics, 1997, 12(4):98—102.

[154] 姜大立，杜文，朱松年.一类二次 0—1 规划模型的遗传算法[J].系统工程，1997，15(4)：21—25.

[155] 隋允康，贾志超，杜家政.非线性 0—1 规划问题的连续化及其遗传算法[J].北京工业大学学报，2008，34(8)：787—791.

[156] 盛红波，孙娟，孙小玲.0—1 多项式背包问题的一种精确算法[J].上海大学学报(自然科学版)，2006，12(4)：389—393.

[157] 贺毅朝，王彦祺，刘建芹.一种适于求解离散问题的二进制粒子群优化算法[J].计算机应用与软件，2007，24(1)：157—159.

[158] 贺一，邱玉辉，刘光远，曾绍华.多维背包问题的禁忌搜索求解[J].计算机科学 2006，33(9)：169—172.

[159] 韩毅，唐加福，牟立峰，等.粒子群算法求解无能力约束生产批量计划问题[J].管理科学学报，2008，11(5)：33—40.

[160] 马良.求解最小比率 TSP 的一个算法[J].系统工程，1998，16(4)：62—65.

[161] Ma L, Cui X L, Yao J. Finding the minimum ratio traveling salesman tour by artificial ants[J]. Journal of Systems Engineering and Electronics, 2003, 14(3):24—27.